GoMAvatar: Efficient Animatable Human Modeling from Monocular Video Using Gaussians-on-Mesh

Supplementary Material

This supplementary material is organized as follows:

1. Sec. A provides the detailed derivation of Gaussians' local-to-world transformation;
2. Sec. B details the whole inference pipeline;
3. Sec. C shows implementation details.
4. Sec. D shows additional results including the quantitative results broken down per scene and additional ablation studies.
5. Sec. E showcases failure cases in our approach.

A. Derivation of Gaussians' local-to-world transformation

As described in Sec. 3.1 and Sec. 3.2, we define the rotation $r_{\theta,j}$ and the scale $s_{\theta,j}$ in the triangle's local coordinates. In order to render with Gaussian splatting, we transform the local Gaussians to the world coordinates. Specifically, the mean is transformed to the centroid of the triangle (Eq. (8)) and the covariance matrix is transformed by the local-to-world transformation matrix A_j (Eq. (9)). We now show the detailed derivation.

Given a face and its associated local properties $f_{\theta,j} = (r_{\theta,j}, s_{\theta,j}, c_{\theta,j}, \{\Delta_{j,k}\}_{k=1}^3)$, we want to compute its Gaussian in the world coordinates $G_j = \mathcal{N}(\mu_j, \Sigma_j)$.

The Gaussian in the triangle's local coordinate is

$$\hat{G}_j = \mathcal{N}(\mathbf{0}, \hat{\Sigma}_j), \text{ where } \hat{\Sigma}_j = R_j S_j S_j^T R_j^T. \tag{16}$$

Here, R_j and S_j are the matrix form of $r_{\theta,j}$ and $s_{\theta,j}$ respectively. Then, we define a transformation from the triangle's local coordinates to world coordinates:

$$f(x) = A_j x + b_j, \tag{17}$$

where $A_j \in \mathbb{R}^{3\times3}$ and $b_j \in \mathbb{R}^3$. Therefore, the mean and covariance of the Gaussian in the world coordinates are

$$\mu_j = b_j, \tag{18}$$
$$\Sigma_j = A_j \hat{\Sigma}_j A_j^T. \tag{19}$$

Local-to-world affine transformation $f(x) = A_j x + b_j$. The goal of the local-to-world affine transformation is to move and reshape the Gaussian based on the location and shape of the triangle face. The world Gaussian's centroid (Eq. (18)) is put at the centroid of the triangle face, i.e.,

$$b_j = \frac{1}{3}\sum_{k=1}^{3} p_{\Delta_{j,k}}. \tag{20}$$

Here, $\{\Delta_{j,k}\}_{k=1}^3$ are the three indices of the vertices on the j-th triangle and hence $\{p_{\Delta_{j,k}}\}_{k=1}^3$ are the coordinates of the vertices.

The matrix $A_j = [a_{j,1}, a_{j,2}, a_{j,3}]$ takes care of the world Gaussian's shape deformation. We use $a_{j,k}$, $k \in \{1,2,3\}$ to denote the three columns of matrix A_j. Our design of A_j is inspired by the Steiner ellipse of a triangle, the unique ellipse that has the maximum area of any ellipse. Specifically, we define $a_{j,1}$ and $a_{j,2}$ as the two semi-axes of the Steiner ellipse:

$$a_{j,1} = \overrightarrow{b_j p_{\Delta_{j,3}}} \cos t_0 + \frac{1}{\sqrt{3}} \overrightarrow{p_{\Delta_{j,1}} p_{\Delta_{j,2}}} \sin t_0, \tag{21}$$

$$a_{j,2} = \overrightarrow{b_j p_{\Delta_{j,3}}} \cos(t_0 + \frac{\pi}{2}) + \frac{1}{\sqrt{3}} \overrightarrow{p_{\Delta_{j,1}} p_{\Delta_{j,2}}} \sin(t_0 + \frac{\pi}{2}), \tag{22}$$

where

$$t_0 = \frac{1}{2}\arctan \frac{\frac{2}{\sqrt{3}} \overrightarrow{b_j p_{\Delta_{j,3}}} \cdot \overrightarrow{p_{\Delta_{j,1}} p_{\Delta_{j,2}}}}{\overrightarrow{b_j p_{\Delta_{j,3}}}^2 - \frac{1}{3}\overrightarrow{p_{\Delta_{j,1}} p_{\Delta_{j,2}}}^2}. \tag{23}$$

The third column $a_{j,3}$ is defined along the normal vector of the triangle face:

$$a_{j,3} = \epsilon \cdot \text{normalize}(a_{j,1} \times a_{j,2}). \tag{24}$$

We multiply the normal vector with ϵ to make sure the ellipsoid is thin along the surface normal. We set $\epsilon = 1\text{e}^{-3}$ in our experiments.

Given the derivation above, when the local rotation $r_{\theta,j}$ is zero and the local scale $s_{\theta,j}$ is one, i.e, $\hat{\Sigma}_j = \mathbf{I}$, the projection of the ellipsoid $\{x : (x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j) = 1\}$ on the triangle recovers the Steiner ellipse, as shown in Fig. 3.

B. Inference Pipeline

We present our inference pipeline including the modules and key inputs and outputs in Fig. 8.

C. Implementation Details

Architecture details. **1)** $\texttt{GoM}_\theta^c$ in Eq. (3) is initialized with the SMPL mesh [43] under the canonical T-pose; **2)** $\texttt{Shading}_\theta$ in Eq. (10) is a 4-layer MLP network with 128 channels; **3)** $\texttt{NRDeformer}_\theta$ in Eq. (12) is a 7-layer MLP network with 128 channels; **4)** $\texttt{PoseRefiner}_\theta$ in Eq. (13) is a 5-layer MLP network with 256 channels. The detailed architecture of $\texttt{Shading}_\theta$, $\texttt{NRDeformer}_\theta$ and $\texttt{PoseRefiner}_\theta$ are shown in Fig. 9.

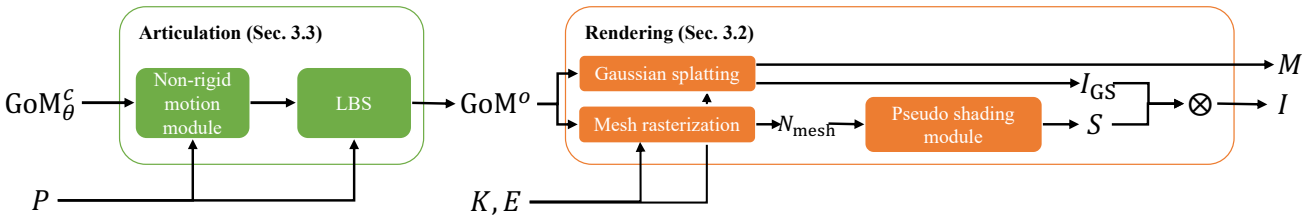

Figure 8. **Inference pipeline.** Our inference pipeline has two stages: 1) **Articulation**: This stage takes the Gaussians-on-Mesh (GoM) representation in the canonical space, denoted as GoM_θ^c, and the human pose P as input. Utilizing the non-rigid motion module and linear blend skinning (LBS), it produces the transformed GoM representation in the observation space, referred to as GoM^o. 2) **Rendering**: In this stage, the transformed GoM^o, along with the camera intrinsic parameters K and extrinsic parameters E, are employed as inputs. It adopts the Gaussian splatting to generate the pseudo albedo map I_{GS} and the subject mask M. Meanwhile, through mesh rasterization, it produces the normal map N_{mesh} which is then fed into the pseudo shading module to output the pseudo shading map S. The final RGB image I is then obtained by multiplying I_{GS} with S.

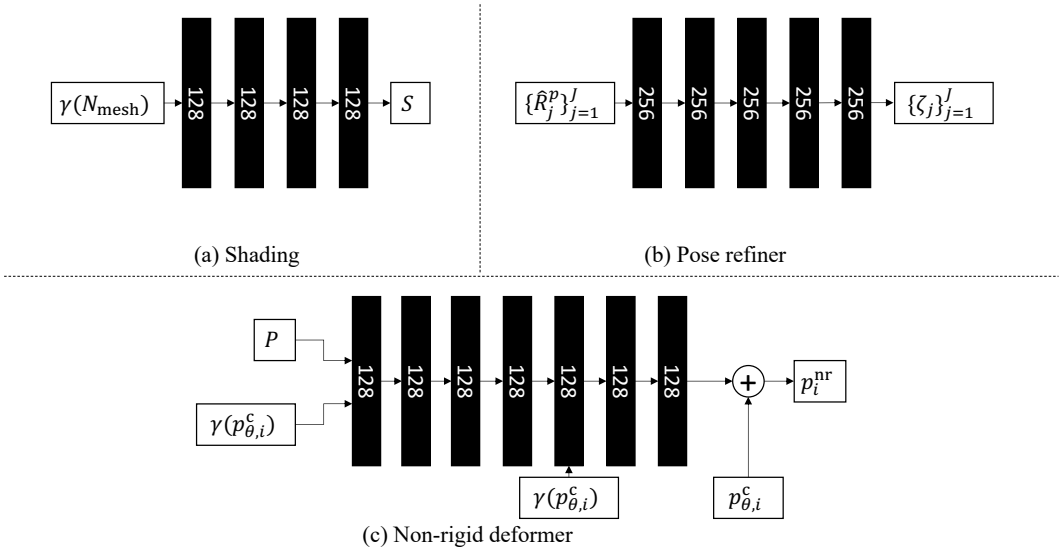

Figure 9. **Detailed architectures of (a) $\texttt{Shading}_\theta$, (b) $\texttt{PoseRefiner}_\theta$ and (c) $\texttt{NRDeformer}_\theta$.**

Training details. We use Adam optimizer [32] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. On ZJU-MoCap, We train the model for 300K iterations. We set the learning rate of $\texttt{PoseRefiner}_\theta$ to 5e$-$5. The learning rate of the rest of the model is 5e$-$4. We kick off the training of $\texttt{PoseRefiner}_\theta$ and $\texttt{NRDeformer}_\theta$ after 100K and 150K iterations respectively. For $\texttt{NRDeformer}_\theta$, we follow HumanNeRF [70] to adopt a HanW window during training. We set $\alpha_{\text{lpips}} = 1.0$ $\alpha_M = 5.0$, $\alpha_{\text{reg}} = 1.0$ in Eq. (14), and $\alpha_{\text{lap}} = 10.0$, $\alpha_{\text{normal}} = 0.1$, $\alpha_{\text{color}} = 0.05$ in Eq. (15). We subdivide the GoM after 50K iterations. On PeopleSnapshot, we train the model for 200K iterations and kick off the training of $\texttt{NRDeformer}_\theta$ after 100K iterations. We subdivide GoM once after 10K iterations. We do not refine training poses with $\texttt{PoseRefiner}_\theta$ following InstantAvatar [26]. On in-the-wild Youtube videos, since the poses are predicted and less accurate, we kick off the training of $\texttt{PoseRefiner}_\theta$ at the start of the training process while keeping all other hyperparameters the same as ZJU-MoCap.

D. Additional Results

D.1. Quantitative Results of Per-scene Breakdown

We show the per-scene PSNR, SSIM and LPIPS* on the ZJU-MoCap dataset in Tab. 6 and Tab. 7. The per-scene breakdown results on PeopleSnapshot is shown in Tab. 8.

D.2. Qualitative Results on PeopleSnapshot

We conduct a qualitative comparison on PeopleSnapshot dataset in Fig. 10. As shown below, we better capture textures better compared to InstantAvatar. Meanwhile, our approach can capture fine details in geometry, such as wrinkles.

	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
	Subject 377			Subject 386			Subject 387		
Neural Body	29.08	0.9679	41.17	29.76	0.9647	46.96	26.84	0.9535	60.82
HumanNeRF	29.79	0.9714	28.49	32.10	0.9642	41.84	28.11	0.9625	37.46
MonoHuman	30.46	0.9781	20.91	32.99	0.9756	30.97	28.40	0.9639	35.06
GoMAvatar (Ours)	30.60	0.9768	23.91	32.97	0.9752	30.36	28.34	0.9635	36.30
	Subject 392			Subject 393			Subject 394		
Neural Body	29.49	0.9640	51.06	28.50	0.9591	57.07	28.65	0.9572	55.78
HumanNeRF	30.20	0.9633	40.06	28.16	0.9577	40.85	29.28	0.9557	41.97
MonoHuman	30.98	0.9711	30.80	28.54	0.9620	34.97	30.21	0.9642	32.80
GoMAvatar (Ours)	31.04	0.9708	33.25	28.80	0.9622	37.77	30.44	0.9646	33.56

Table 6. **Per-scene breakdown in novel view synthesis on ZJU-MoCap dataset.**

	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
	Subject 377			Subject 386			Subject 387		
Neural Body	29.29	0.9693	39.40	30.71	0.9661	45.89	26.36	0.9520	62.21
HumanNeRF	29.91	0.9755	23.87	32.62	0.9672	39.36	28.01	0.9634	35.27
MonoHuman	30.77	0.9787	21.67	32.97	0.9733	32.73	27.93	0.9633	33.45
GoMAvatar (Ours)	30.68	0.9776	23.41	32.86	0.9737	32.25	28.18	0.9626	36.43
	Subject 392			Subject 393			Subject 394		
Neural Body	28.97	0.9615	57.03	27.82	0.9577	59.24	28.09	0.9557	59.66
HumanNeRF	30.95	0.9687	34.23	28.43	0.9609	36.26	28.52	0.9573	39.75
MonoHuman	31.24	0.9715	31.04	28.46	0.9622	34.24	28.94	0.9612	35.90
GoMAvatar (Ours)	31.44	0.9716	33.20	29.09	0.9635	36.02	29.79	0.9638	33.00

Table 7. **Per-scene breakdown in novel pose synthesis on ZJU-MoCap dataset.**

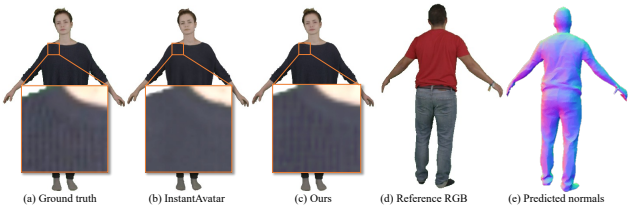

Figure 10. **Qualitative results on PeopleSnapshot dataset.** On the left side, we conduct a qualitative comparison to InstantAvatar. We also show the geometry by rendering the surface normals on the right side.

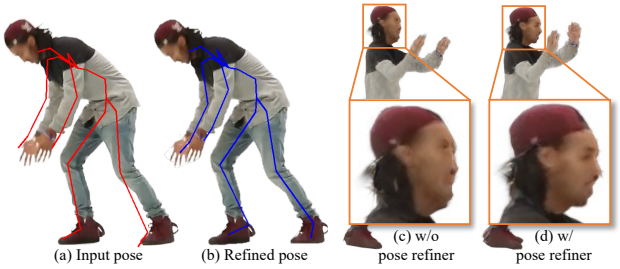

Figure 11. **Robustness to SMPL accuracy.** Our method is robust to SMPL prediction. This can be seen in in-the-wild videos. (a) Predicted poses have errors. (b) Pose refinement improves erroneous SMPL poses, which is crucial for in-the-wild videos (c, d).

D.3. Sensitivity to SMPL Accuracy

Our approach takes the human poses in the input frames as inputs. The human poses are provided in ZJU-MoCap dataset and PeopleSnapshot dataset, while we predict the poses with PARE [33] for in-the-wild Youtube videos.

The robustness to SMPL prediction can be seen in in-the-wild videos. In Fig. 11(a), we show that there are errors in pose prediction in in-the-wild videos. However, the pose refinement improves erroneous SMPL poses (Fig. 11(b)). The pose refinement is crucial for rendering in in-the-wild videos, which can be seen in Fig. 11(c, d). Without the pose refinement, the approach fails to render the correct shape of the human face. This issue is solved when equipped with the pose refinement.

In Tab. 9, we quantitatively assess sensitivity to SMPL accuracy on ZJU-MoCap, comparing the original less-accurate SMPL poses to refined versions from Instant-NVR [16]. Hence, refining SMPL pose improves rendering

	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
	m3c			m4c		
Anim-NeRF	29.37	0.9703	0.0168	28.37	0.9605	0.0268
InstantAvatar	29.65	0.9730	0.0192	27.97	0.9649	0.0346
GoMAvatar (Ours)	31.74	0.9793	0.0187	29.78	0.9738	0.0282
	f3c			f4c		
Anim-NeRF	28.91	0.9743	0.0215	28.90	0.9678	0.0174
InstantAvatar	27.90	0.9722	0.0249	28.92	0.9692	0.0180
GoMAvatar (Ours)	29.83	0.9758	0.0209	31.38	0.9780	0.0174

Table 8. **Per-scene breakdown in novel view synthesis on PeopleSnapshot dataset.**

	PSNR ↑	SSIM ↑	LPIPS*↓	CD ↓	NC ↑
Original	30.37	0.9689	32.53	2.8364	0.6201
Refined	30.86	0.9709	30.91	2.3377	0.6307

Table 9. **Quantitative evaluation about sensitivity to SMPL accuracy.** We test our approach with two versions of SMPL poses on ZJU-MoCap dataset. "Original" refers to the poses provided in the original ZJU-MoCap dataset, which is less accurate. "Refined" refers to the improved version from InstantNVR [16].

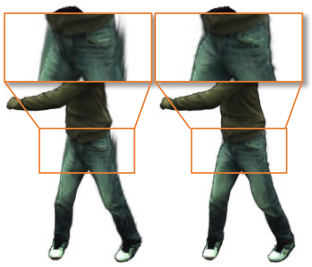

Figure 12. **Qualitative comparison between GoM and Gaussians only.**

and geometry quality, but our method will *not* fail without.

D.4. Ablation on Canonical Representations

In Sec.4.4, we conduct a quantitative comparison of the GoM presentation and 3D Gaussians alone. Here, we show the comparison to *Gaussians only* qualitatively in Fig. 12. *Gaussians only* yield severe artifacts on the boundary while our method attains a sharp boundary.

E. Failure Cases

We present two failure cases of our approach:

1. Our approach, along with other state-of-the-art methods optimized on a per-scene basis, lacks the ability to hallucinate unseen regions. This limitation can be observed in the failure for Subject 386 in the ZJU-MoCap dataset, as shown in Fig. 13(a). In subject 386, the training frames do not cover the front view of the person. Consequently, all methods fail to generate a valid rendering from this unobserved perspective.
2. As we associate Gaussians with the mesh in the Gaussians-on-Mesh representation, we sometimes cannot handle significant topology changes. One example is the white belt on the shorts in Subject 377 (Fig. 13(b)), which dynamically shifts with the person's movement. Interestingly, when fitting clothes with different topologies from SMPL, such as dresses, our model can self-deform to fit the shapes and yield plausible novel-view

(a) (b)

Figure 13. **Failure cases.**

Figure 14. **Novel view synthesis on subjects in dresses.**

renderings, even though the topology does not change. This can be seen in Fig. 14. Addressing topology changes may require a pose-dependent topology update, which we leave to future work.