RGBAvatar: Reduced Gaussian Blendshapes for Online Modeling of Head Avatars

Supplementary Material

In this supplementary document, we provide additional details about our method, experiments, and results. In Sec. 7, we elaborate on the Gaussian transformation calculations, the online frame processing algorithm, and implementation details. Section 8 includes further ablation studies, experiments and limitations. Finally, in Sec. 9, we showcase online reconstruction results and provide additional qualitative comparisons. We also encourage viewing our supplemental video for a comprehensive demonstration, including a real-time online reconstruction showcase.

7. Method Details

7.1. Gaussian Transformation

To incorporate geometric priors from the 3DMM mesh, we define Gaussian attributes in tangent space $\mathcal{T}$. For rendering, these Gaussians are transformed into deformed space $\mathcal{D}$ based on mesh deformation, as illustrated in Fig. 12. For a mesh M^θ under expression parameters θ, the transformation involves computing the TBN matrix $\mathbf{R}$ for each triangle (see Algorithm 1). Gaussian position $\mathbf{x}^\mathcal{T}$ and rotation $\mathbf{q}^\mathcal{T}$ are transformed to $\mathcal{D}$ as follows:

$$\mathbf{x}^\mathcal{D} = \mathbf{R}\mathbf{x}^\mathcal{T} + \mathbf{t},\ \mathbf{q}^\mathcal{D} = \mathbf{R}\mathbf{q}^\mathcal{T}, \tag{4}$$

where $\mathbf{t}$ is the translation vector, computed via barycentric interpolation using the triangle vertices and the Gaussian's UV coordinates. The UV coordinates are initialized by sampling from the UV map and remain fixed during optimization.

7.2. Online Frame Processing

For online reconstruction, each incoming frame is processed sequentially, as shown in Algorithm 2. We first use a real-time 3DMM tracker [13] to extract 3DMM parameters θ from the incoming frame I_i and compute the corresponding mesh M_i. The data sample $D_i = I_i, \theta_i, M_i$ is then assembled.

If the local sampling pool $\mathcal{M}_l$ is full, the last sample D_j is removed and appended to the global sampling pool $\mathcal{M}_g$ with probability $\frac{|\mathcal{M}_g|}{j}$. Finally, D_i is added to $\mathcal{M}_l$. The local pool $\mathcal{M}_l$ operates in a FIFO manner, while the global pool $\mathcal{M}_g$ uses Reservoir Sampling [42]. The avatar model is optimized using batched data sampled from both pools.

7.3. Implementation Details

Our method is implemented in PyTorch, with the Gaussian Transformation, Linear Blending, and Color Initialization modules written in CUDA as PyTorch extensions. The batch-parallel Gaussian rasterizer builds upon the 3DGS renderer [27].

Algorithm 1 TBN Matrix Calculation

Input: Triangle vertices $\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}$ and texture coordinates $\mathbf{uv_0} = (u_0, v_0), \mathbf{uv_1} = (u_1, v_1), \mathbf{uv_2} = (u_2, v_2)$
Output: TBN matrix $\mathbf{R}$

$$\mathbf{M} \leftarrow \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix}$$
$$\mathbf{e_1} \leftarrow \mathbf{v_1} - \mathbf{v_0}$$
$$\mathbf{e_2} \leftarrow \mathbf{v_2} - \mathbf{v_0}$$
$$\mathbf{N} \leftarrow \frac{\mathbf{e_1} \times \mathbf{e_2}}{||\mathbf{e_1} \times \mathbf{e_2}||}$$
$$\begin{bmatrix} \mathbf{T} \\ \mathbf{B} \end{bmatrix} \leftarrow \mathbf{M^{-1}} \begin{bmatrix} \mathbf{e_1} \\ \mathbf{e_2} \end{bmatrix}$$
$$\mathbf{R} \leftarrow \begin{bmatrix} \mathbf{T} & \mathbf{B} & \mathbf{N} \end{bmatrix}$$

We use the Adam optimizer with a batch size of 10. Learning rates for $\mathbf{x}$, α, $\mathbf{s}$, $\mathbf{q}$, and $\mathbf{c}$ are set to 0.0008, 0.25, 0.025, 0.005, and 0.0125, respectively. For blendshape parameters $\Delta\mathbf{x}$, $\Delta\mathbf{q}$, and $\Delta\mathbf{c}$, the learning rates are scaled by 0.05, 0.5, and 0.5, relative to $\mathbf{x}$, $\mathbf{q}$, and $\mathbf{c}$. Blendshapes are not applied to Gaussian opacity or scaling. We only use RGB color for appearance modeling. Besides, we apply activation functions after the linear blending to ensure the validity of Gaussian parameters, following GaussianBlendshapes [33].

For the offline setting, we optimize our model for 5000 steps. In the online setting, to enable a more reasonable comparison with the offline setting to evaluate our online quality, we simulate a real-time on-the-fly setting by streaming pre-processed video frames and FLAME parameters at 25 FPS. The training steps depend on the length of the video stream, *i.e.* 2-minute video requires about 75k steps (120s×630 training steps/s). We also implement a real-time reconstruction pipeline using a real-time tracker [13] and the FaceWarehouse 3DMM model [12]. All experiments were conducted on a single NVIDIA RTX 3090.

The MLP $\mathcal{F}$ consists of three layers with a 128-dimensional latent feature and is optimized with a learning rate of 0.001. Following GaussianAvatars [37], we add 60 additional triangle faces to model teeth. The UV map resolution is initialized at 256, resulting in approximately 60k Gaussians.

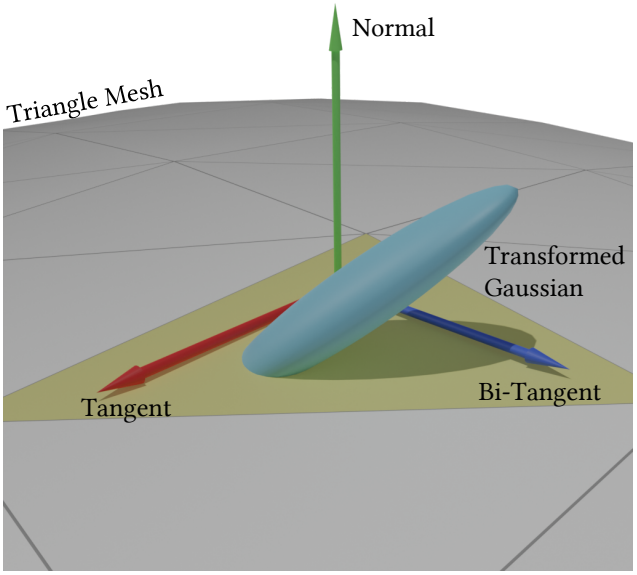

Figure 12. **Gaussian Transformation.** The corresponding triangle face and the transformed Gaussian are illustrated in yellow and sky blue, respectively.

Algorithm 2 Processing Incoming Frame

Input: Input frame I_i, local sampling pool $\mathcal{M}_l$, global sampling pool $\mathcal{M}_g$
Output: Updated $\mathcal{M}_l$ and $\mathcal{M}_g$

$\theta_i \leftarrow \text{tracker}(I_i)$
$M^\theta \leftarrow \text{compute\_3DMM\_mesh}(\theta_i)$
$D_i \leftarrow \{I_i, \theta_i, M^\theta\}$
if $\mathcal{M}_l$ is full **then**
  $D_j \leftarrow \mathcal{M}_l.\text{pop\_front}()$
  if $\mathcal{M}_g$ is full **then** ▷ Reservoir Sampling
    $k \leftarrow \text{randint}(0, j)$
    if $k < |M_g|$ **then** ▷ $|\mathcal{M}_g|$ is the size of $\mathcal{M}_g$
      $\mathcal{M}_g[k] \leftarrow D_j$
    end if
  else
    $\mathcal{M}_g.\text{append}(D_j)$
  end if
end if
$\mathcal{M}_l.\text{append}(D_i)$

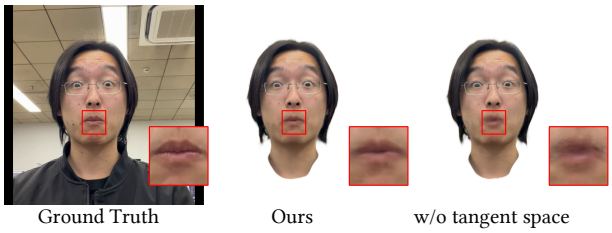

Figure 13. **Impact of tangent space.** Without geometry prior provided by mesh deformation, the model delivers blurred results.

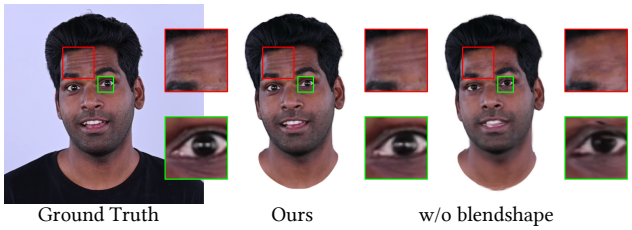

Figure 14. **Impact of blendshape.** Without blendshapes, the model lacks representation capacity to model wrinkle-level details.

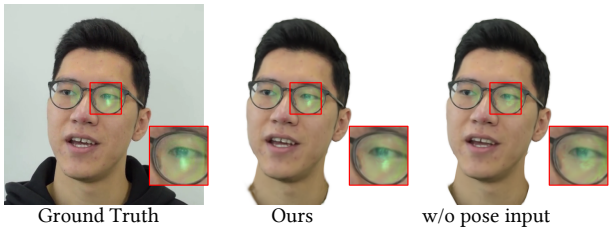

Figure 15. **Impact of pose input.** Without joint pose input, the model lacks information represents pose-relevant appearance, such as eyeglass reflection.

Method	PSNR	SSIM	LPIPS
w/o tangent space	31.01	0.9513	0.0825
w/o blendshape	29.12	0.9402	0.0920
w/o pose input	30.79	0.9533	0.796
Ours (linear)	**31.38**	0.9556	0.772
Ours	31.19	**0.9565**	**0.737**

Table 5. **Ablation study on model design.** Our default setting outperforms other variants. Experiments are conducted on INSTA dataset.

8. Additional Experiments

8.1. Ablation on Model Design

We conducted ablation studies to evaluate our model design, with quantitative results shown in Tab. 5. Our default design consistently outperforms other variants.

First, we tested a variant without defining Gaussians in tangent space (*w/o tangent space*), where Gaussians are transformed using joint poses instead of mesh deformation, similar to GaussianBlendShapes [33]. Since our reduced blendshapes are subject-adaptive, initializing with generic FLAME blendshapes is infeasible. Consequently, the joint pose rotations provide insufficient geometric priors, leading to suboptimal Gaussian transformations. As shown in Fig. 13, this variant relies on color composition to fit subtle

Method	PSNR↑	SSIM↑	LPIPS↓	e_{orth}
Orthogonal loss	30.91	0.9499	0.0890	9.164×10^{-4}
QR decomposition	30.26	0.9462	0.0912	9.558×10^{-8}
Schmidt orthogonalization	30.75	0.9492	0.0892	4.157×10^{-8}
Ours (not orthogonal)	**31.16**	**0.9551**	**0.0725**	3.173

Table 6. **Investigation of basis orthogonality**. Experiments are conducted on INSTA dataset.

deformations (e.g., around the mouth), resulting in blurred self-reenactment results.

Second, we evaluated a variant without blendshapes (*w/o blendshape*), where only a single set of base Gaussians is attached to the 3DMM mesh, similar to SplattingAvatar [39] and GaussianAvatars [37]. This configuration lacks the capacity to model finer details, such as wrinkles or eyeball reflections, as shown in Fig. 14.

Third, we ablated the FLAME parameters provided to the MLP, which are mapped to reduced blendshape weights. Our default setting (*Ours*) includes both expression coefficients and joint poses. When joint poses are excluded (*w/o pose input*), the model fails to represent pose-dependent appearance features, such as eyeglass reflections (Fig. 15).

Finally, we replaced the MLP in our model with a simple linear projection (*Ours (linear)*). As shown in Tab. 5, while this variant achieves slightly higher PSNR, it performs worse on SSIM and LPIPS metrics.

8.2. Ablation on Online Reconstruction Strategy

We further analyzed the importance of our global and local sampling strategy for online reconstruction.

Fig. 16 compares the per-frame L1 loss with (left) and without (right) global sampling. The blue line shows the L1 loss after online training, while the orange line indicates the minimum L1 loss during training. The gap between these lines reflects the extent of "forgetting." Our sampling strategy effectively mitigates this issue.

Additionally, we examined the impact of pool size on reconstruction quality (Tab. 7). The results show that varying pool sizes has minimal effect. Based on empirical observations, we set the local pool size to $|\mathcal{M}_l| = 150$ and the global pool size to $|\mathcal{M}_g| = 1000$.

8.3. Investigation of Basis Orthogonality

The blendshape bases of FLAME are constructed using PCA, making them inherently orthogonal. However, our reduced blendshape bases break this orthogonality during training. Therefore, we examine whether maintaining orthogonality is necessary. First, we formally define the orthogonality of the 20 reduced blendshape bases as

$$e_{orth} = ||VV^T - I||_2, \tag{5}$$

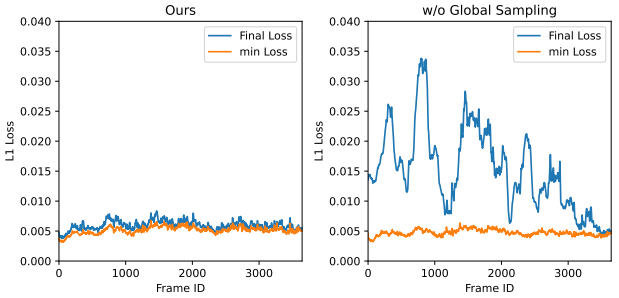

Figure 16. **Impact of global sampling on online reconstruction.** Left: per frame L1 loss with our sampling strategy. Right: per frame L1 loss without gloabal sampling.

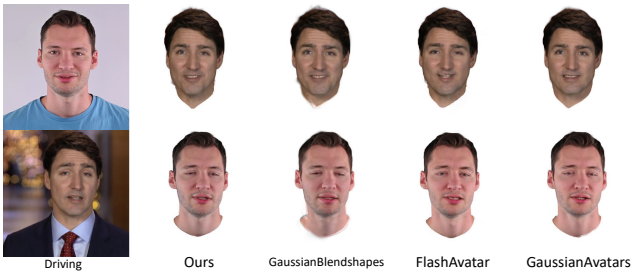

Figure 17. **Qualitative comparisons of cross-id reenactment.**

Method	PSNR	SSIM	LPIPS
$\|\mathcal{M}_l\| = 150, \|\mathcal{M}_l\| = 500$	30.90	0.9537	0.0758
$\|\mathcal{M}_l\| = 150, \|\mathcal{M}_l\| = 1000$	31.04	0.9543	0.0758
$\|\mathcal{M}_l\| = 150, \|\mathcal{M}_l\| = 1500$	30.80	0.9544	0.0755
$\|\mathcal{M}_l\| = 150, \|\mathcal{M}_l\| = 2000$	**31.06**	0.9546	0.0759
$\|\mathcal{M}_l\| = 50, \|\mathcal{M}_l\| = 1000$	30.94	0.9542	0.0763
$\|\mathcal{M}_l\| = 100, \|\mathcal{M}_l\| = 1000$	30.90	**0.9548**	0.0755
$\|\mathcal{M}_l\| = 200, \|\mathcal{M}_l\| = 1000$	30.97	0.9546	**0.0754**

Table 7. **Ablation study on sampling pool size of online reconstruction.** Experiments are conducted on INSTA dataset.

where $V \in \mathbb{R}^{20 \times d}$ represents the concatenation of the normalized 20 bases. The bases are perfectly orthogonal when $e_{orth} = 0$, although minor numerical errors may arise in practice. Then we conduct three different experiments to ensure orthogonality: we add 1) an *orthogonal loss* with

Method	AED↓	APD↓	CSIM↑
MonoGaussianAvatar	10.9268	0.1406	0.7618
GaussianAvatars	9.9394	<u>0.1167</u>	0.7858
FlashAvatar	9.9210	0.1250	**0.7952**
GaussianBlendShapes	**9.1241**	**0.1158**	0.7866
Ours	<u>9.2683</u>	0.1179	<u>0.7868</u>

Table 8. **Quantitative comparisons of cross-id reenactment.**

Figure 18. **Limitation of tracking error.**

loss weight $\lambda_{orth} = 10$. we perform 2) *QR decomposition* or 3) *Schmidt orthogonalization*, both every 1k training steps. As shown in Tab. 6, our method achieves the best result. Since our objective is to reconstruct a photorealistic Gaussian avatar, it is not necessary for the blendshape bases to be orthogonal.

8.4. Comparison of Generalization Ability

We futher evaluate 10 cross-identity reenactment video sequences using standard metrics from the image animation literature [15]: average expression distance(AED), average pose distance(APD), and identity similarity(CSIM). As shown in Tab. 8 and Fig. 17, our method yields comparable results to other Gaussian-based methods.

8.5. Limitation of Tracking Error

Our method, along with others, exhibits artifacts caused by tracking errors. We demonstrate a common tracking error that occurs with a large side head pose in Fig. 18.

9. Additional Results

We present the online reconstruction results in Fig. 19, demonstrating comparable quality to the offline setting. Additional qualitative comparisons with other methods are provided in Fig. 20.

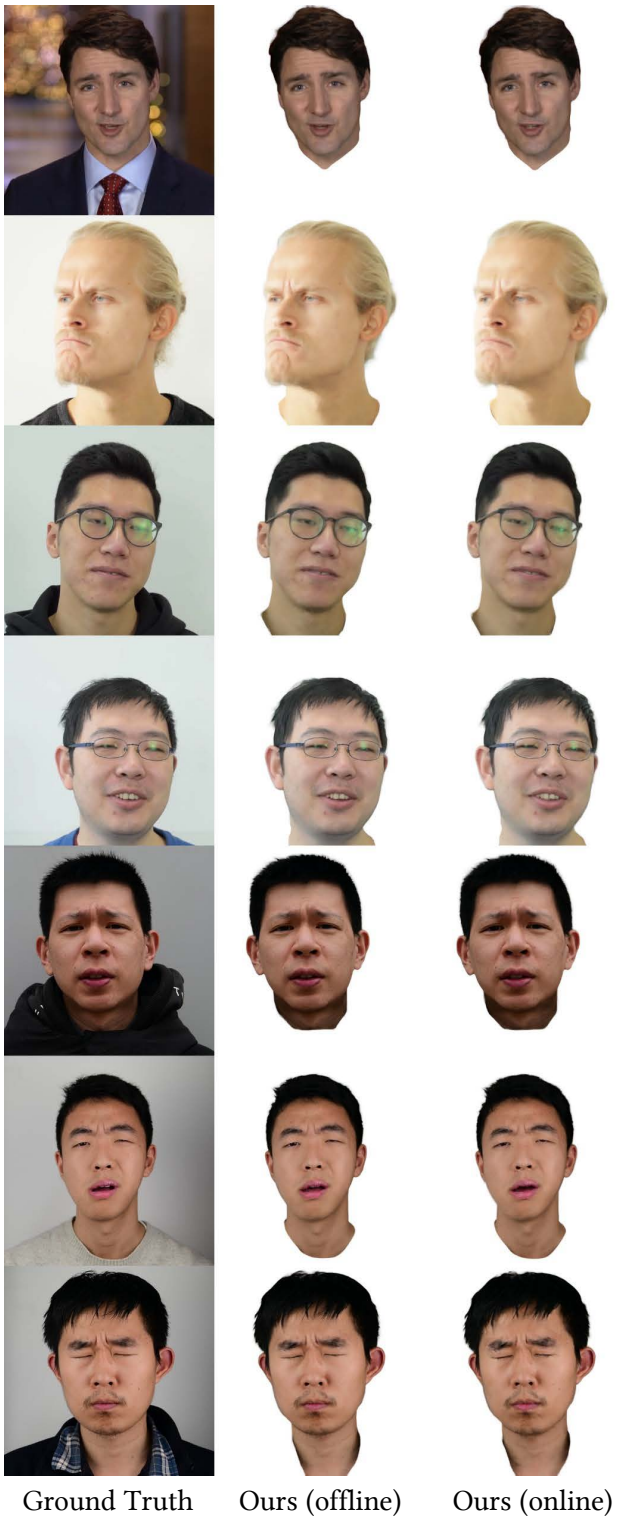

Figure 19. **Results of online reconstruction.** Our online reconstruction strategy achieves comparable quality to offline setting.

Figure 20. **Additional Qualitative comparisons.**