SqueezeMe: Mobile-Ready Distillation of Gaussian Full-Body Avatars

FORREST IANDOLA, Meta Reality Labs, USA
STANISLAV PIDHORSKYI, Meta Codec Avatars Lab, USA
IGOR SANTESTEBAN, Meta Codec Avatars Lab, USA
DIVAM GUPTA, Meta Codec Avatars Lab, USA
ANUJ PAHUJA, Meta Codec Avatars Lab, USA
NEMANJA BARTOLOVIC, Meta Codec Avatars Lab, Switzerland
FRANK YU, Meta Codec Avatars Lab, USA
EMANUEL GARBIN, Meta Codec Avatars Lab, Israel
TOMAS SIMON, Meta Codec Avatars Lab, USA
SHUNSUKE SAITO, Meta Codec Avatars Lab, USA

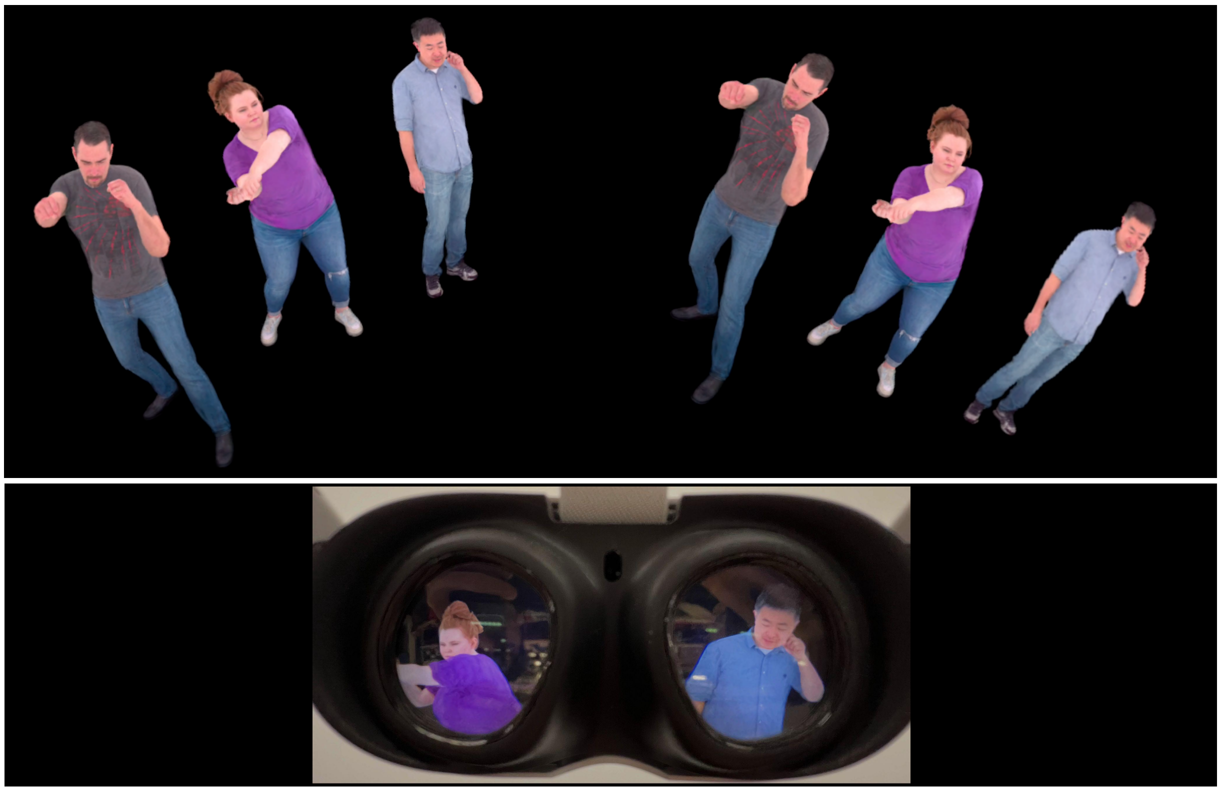


Fig. 1. **With SqueezeMe, we simultaneously run 3 Gaussian avatars locally on a Meta Quest 3 VR headset.** Upper: Stereo view in VR. Lower: View into the VR headset lenses.

Gaussian-based human avatars have achieved an unprecedented level of visual fidelity. However, existing approaches based on high-capacity neural networks typically require a desktop GPU to achieve real-time performance for a single avatar, and it remains non-trivial to animate and render such avatars on mobile devices including a standalone VR headset due to substantially limited memory and computational bandwidth. In this paper, we present SqueezeMe, a simple and highly effective framework to convert high-fidelity 3D Gaussian full-body avatars into a lightweight representation that supports both animation and rendering with mobile-grade compute. Our key observation is that the decoding of pose-dependent Gaussian attributes from a neural network creates non-negligible memory and computational overhead. Inspired by blendshapes and linear pose correctives widely used in Computer Graphics, we address this by distilling the pose correctives learned with neural networks into linear layers. Moreover, we further reduce the parameters by sharing the correctives among nearby Gaussians. Combining them with a custom splatting pipeline based on Vulkan, we achieve, for the

.

first time, simultaneous animation and rendering of 3 Gaussian avatars in real-time (72 FPS) on a Meta Quest 3 VR headset. Demo videos are available at https://forresti.github.io/squeezeme.

CCS Concepts: • **Computing methodologies → Reconstruction**; **Animation**.

Additional Key Words and Phrases: 3D Avatar Creation, Neural Rendering

1 Introduction

Augmented Reality (AR) and Virtual Reality (VR) have shown great promise in delivering immersive experiences that blur the lines between the physical and virtual worlds. At the heart of these experiences lies photorealistic full-body avatars, whose animation and appearance are indistinguishable from reality. Such avatars enable a wide range of applications, including telepresence, virtual try-on, and immersive gaming. To make such technology accessible to a broader population, standalone AR/VR devices with lightweight form factors are required, necessitating a highly restrictive computational budget. Our goal is to enable the animation and rendering of photorealistic avatars on mobile-grade compute, thereby paving the way for widespread adoption of authentic digital humans in AR/VR.

Recent advances in accelerating the rendering of photorealistic 3D scenes based on NeRF [Chen et al. 2023a; Reiser et al. 2023, 2024] and 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023; Niedermayr et al. 2024] have shown promising results. Moreover, real-time rendering of volumetric videos on mobile devices has become possible [Wang et al. 2024b]. However, these approaches are not directly applicable to avatars, because avatars require support for not only static scenes or the playback of pre-recorded sequences, but also novel animations computed on-the-fly for real-time driving of avatars [Bagautdinov et al. 2021]. Existing approaches that can achieve photorealism rely on high-capacity neural networks to decode non-rigid correctives given a driving signal such as body pose parameters [Li et al. 2024; Svitov et al. 2024; Wang et al. 2022]. Running such a decoder for every frame creates non-negligible memory and computational overhead, hindering the deployment of photorealistic avatars on device.

To address this, we present SqueezeMe, a novel approach to distilling photorealistic avatars based on 3D Gaussians into a lightweight representation that is ready for animation and rendering on mobile devices. Inspired by pose-dependent linear correctives, widely used for computer graphics [Loper et al. 2015b], we compute a linear mapping from pose parameters to geometry and appearance parameters of 3D Gaussians, including rotation, displacements, scale, and spherical harmonics coefficients. More specifically, we first train 3D Gaussian avatars with pose-dependent corrective parameters defined on a UV map using a high-capacity convolutional neural network (CNN), which achieves comparable performance with a state-of-the-art Gaussian avatar method [Li et al. 2024] with 5 times fewer Gaussians. Then, we extract key frames with associated Gaussian parameters to ensure the uniform coverage of various poses. We then solve a linear regression from the associated pose parameters to the target correctives.

Although this linear distillation drastically simplifies the computations performed in the decoder, the size of the linear matrix remains relatively large, creating a non-trivial memory overhead for mobile compute. We observe that while static Gaussian parameters need to preserve high-frequency signals to be photorealistic, pose-dependent correctives tend to be low-frequency. Based on this key insight, we further reduce the memory footprint by sharing the correctives among Gaussians that are adjacent on the UV map layout. This allows us to reduce the required memory roughly 16 times, while minimally compromising visual fidelity and high-frequency person-specific details.

During inference, we render the Gaussians with the parameters computed from the linear model and use a custom renderer based on Vulkan. Our experiments demonstrate that our approach allows us to animate and render up to 3 full-body avatars at real-time framerates (72FPS) on a Meta Quest 3 headset. We also carefully evaluate our approach and show that quality degradation by the proposed distillation is minimal.

In summary, our contributions include:

- We present a framework to distill personalized Gaussian full-body avatars into a lightweight representation, enabling simultaneous animation and rendering of 3 avatars in real-time on a standalone VR headset (Quest 3) for the first time.
- We learn a linear mapping from pose parameters to Gaussian corrective values using key frames of a Gaussian avatar based on high-capacity CNN.
- We further reduce the memory footprint by sharing pose-dependent correctives in the UV space with minimal quality degradation.

2 Related Work

2.1 Photorealistic Full-Body Avatars

Creating photrealistic avatars has been a long-standing problem in computer graphics and vision. Earlier works attempt modeling clothed humans by deforming a single template mesh such as SMPL [Loper et al. 2015a] with texture maps [Alldieck et al. 2018a,b; Bagautdinov et al. 2021]. While the mesh-based approaches work well for tight clothing and skin regions, hair is typically difficult to handle due to topological constraints and difficulty in handling opacity. Avatars based on Signed Distance Fields (SDF) [Chen et al. 2024; Guo et al. 2023; Jiang et al. 2022a; Saito et al. 2021; Wang et al. 2022; Zhu et al. 2024] can handle topology changes, but have difficulty handling intricate geometric details such as hair. Volumetric representations such as Neural Volumes [Lombardi et al. 2019] or NeRF [Mildenhall et al. 2021] address the limitation of meshes by explicitly integrating opacity in volumetric rendering equations. These approaches are later expended to human modeling [Jiang et al. 2022b; Li et al. 2022; Liu et al. 2021; Peng et al. 2021a,b; Su et al. 2021; Weng et al. 2020, 2022]. While these volumetric representations are typically slow to render due to expensive evaluation, follow-up works accelerate the rendering for real-time inference by utilizing primitive approximation of volumetric fields with voxels [Chen et al. 2023b; Lombardi et al. 2021; Remelli et al. 2022] or point clouds [Prokudin et al. 2023; Su et al. 2023; Zheng et al. 2023a]. However, these approaches still struggle with modeling detailed shape and appearance. More recently, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] achieves better trade-off for rendering speed and fidelity by approximating radiance fields with anisotropic

Gaussians. Full-body avatars based on 3DGS achieve state-of-the-art performance [Hu et al. 2024; Hu and Liu 2024; Li et al. 2024; Pang et al. 2024; Qian et al. 2024; Zielonka et al. 2023]. While these approaches based on 3DGS achieves real-time animation and rendering on desktop-grade GPUs, decoding pose correctives per frame remains prohibitively slow or intractable to run on mobile compute due to high memory and computation requirements. In contrast, our work proposes the distillation of these high-capacity CNN decoders into lightweight linear layers with significant memory reduction by corrective sharing, enabling the animation and rendering of pose-dependent animatable Gaussian avatars.

2.2 Efficient Volumetric Rendering

Efficient rendering of radiance fields are also extensively studied. Earlier works utilize multiplane image (MPI) [Wizadwongsa et al. 2021] or sparse voxel grids [Hedman et al. 2021] to drastically reduce the number of evaluations per ray. More recent works further reduce the evaluation time to once per ray by extracting the surface mesh [Chen et al. 2023a; Duckworth et al. 2024; Guédon and Lepetit 2024; Reiser et al. 2023; Yariv et al. 2023]. 3DGS rendering can also be accelerated by integer quantization, vector quantization, and hash grids to reduce the size footprint of 3DGS as demonstrated in [Fan et al. 2023; Lee et al. 2024; Niedermayr et al. 2024]. These approaches are only applicable to static scenes, and more recent works support efficient rendering of 4D dynamic scenes using triplane [Wu et al. 2024], voxels [Lin et al. 2023; Wang et al. 2024a], depth peeling [Xu et al. 2024a], and dynamic 3D Gaussians [Sun et al. 2024; Xu et al. 2024b; Yang et al. 2024]. However, they are either non-trivial to stream and render on mobile devices [Sun et al. 2024; Wu et al. 2024; Xu et al. 2024a] or prone to blurry results [Wang et al. 2024a]. V3 [Wang et al. 2024b] achieves mobile streaming of dynamic Gaussian splatting by converting dynamic Gaussians to 2D image sequences and applying video codec. Unlike these methods, which support only playback of pre-recorded sequences, our approach compresses the basis of correctives and effectively supports novel animations of photorealistic avatars with on-the-fly driving signals with mobile-grade compute.

2.3 Efficient Animatable Avatars

Improving the efficiency of rendering and animation of photorealistic avatars is an active research topic in the community. Pixel Codec Avatars [Ma et al. 2021] presents a method to reduce the memory and computation overhead of head avatars by offloading view-dependent texture decoding to per-pixel computation via neural deferred rendering [Thies et al. 2019] with low-resolution latent codes and high-resolution static maps. MoRF utilizes neural deferred shading for full-body avatar animation and rendering [Bashirov et al. 2024]. However, due to the limited compute on mobile, the framerate and resolution are limited to 30 FPS with 640×640 on Qualcomm Snapdragon. Unfortunately, this is insufficient for VR, where we need at least 72 FPS with $2K$ resolution for two eyes. SplattingAvatar [Shao et al. 2024] is a hybrid representation of Gaussian Splats embedded on a tracked mesh, achieving 30 FPS for a single avatar on iPhone 13. However, this approach does not account for pose-dependent correctives, limiting its animation fidelity. The closest to our work are Gaussian Blendshapes [Ma et al. 2024; Yan et al. 2025] and Gaussian Eigen Models [Zielonka et al. 2024], where the linear basis of Gaussian parameters in head avatars is extracted from pretrained CNN decoders. However, we find that simply extracting linear basis is still not sufficient to meet memory and computational requirements on mobile. Moreover, unlike head only avatars, full-body avatars need to explicitly account for pose-dependent deformations during the distillation process. In contrast, we present, for the first time, a complete system to enable real-time animation and rendering of full-body Gaussian avatars that meet requirements for VR rendering.

3 Preliminary

3.1 Gaussian Splatting

3D Gaussian Splatting [Kerbl et al. 2023] is a powerful representation for computer graphics. Each Gaussian is parameterized by several terms including rotation, translation μ, scale σ, and a set of spherical harmonics terms for view-dependent color. The Gaussian's rotation and scale construct a covariance matrix Σ. Each Gaussian also has a density term δ, which defines the opacity at the center of the Gaussian.

Let us view the Gaussians from a specific camera c, defined with focal length (f_x, f_y), translation $t_c = (x_c, y_c, z_c)$, and rotation matrix R_c. The Jacobian of this camera is

$$J = \begin{bmatrix} \frac{f_x}{z_c} & 0 & -\frac{f_x x_c}{z_c^2} \\ 0 & \frac{f_y}{z_c} & -\frac{f_y y_c}{z_c^2} \end{bmatrix}. \tag{1}$$

From the camera perspective, the 2D covariance of the Gaussian is $\Sigma_{proj} = J R_c \Sigma R_c^T J^T$. Let μ_p represent the translation of a Gaussian in pixel space.

After sorting the Gaussians based on their depth and their visibility to each pixel, the Gaussians are rasterized onto an image as follows. The opacity α of a Gaussian k to a pixel located at $\rho = (\rho_x, \rho_y)$ units from the image center is

$$\alpha_k = \delta\, exp(-\frac{1}{2}(\mu_\rho - \rho)^T \Sigma_{proj}^{-1}(\mu_\rho - \rho)). \tag{2}$$

The final color of pixel ρ is computed as

$$C = \sum_{i=1}^{N} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) c_i, \tag{3}$$

where N is the number of Gaussians visible to pixel ρ, and the view-dependent color c_i is computed using spherical harmonics.

4 Method

To produce a mobile-ready 3D Gaussian Avatar, we first train a high-fidelity avatar driven by Linear Blend Skinning (LBS) and nonlinear correctives computed by a 2D convolutional decoder (Section 4.1). This decoder is then distilled into a linear model that can run efficiently on mobile hardware with minimal quality degradation (Section 4.2).

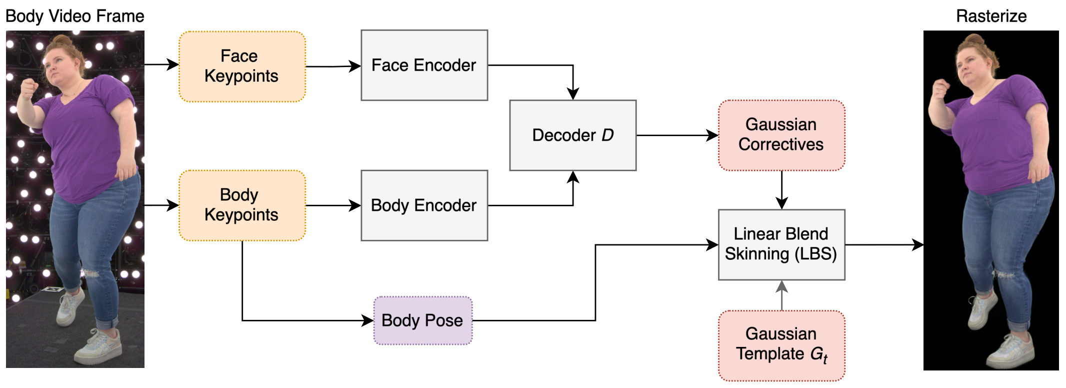


Fig. 2. **System Diagram during training.** This is the configuration we use for training the model in Section 4.1.

4.1 Compact 3D Gaussian Avatar

Our high-fidelity 3D avatar consists of a convolutional decoder that predicts a 2D gaussian map $\mathrm{G} \in \mathbb{R}^{K\times256\times256}$ where each pixel represents a Gaussian and $K = 37$ are the degrees of freedom of each Gaussian. In contrast to Animatable Gaussians [Li et al. 2024], which uses front/back orthogonal projection with many unused pixels, we project Gaussian properties to the UV map of a skinned body mesh. This results in a more compact representation where we can achieve higher Gaussian counts with smaller 2D maps.

Initialization. To ease the training, we use a personalized skinned mesh as our starting point. We obtain this personalized mesh for each identity using LBS-based surface tracking [Gall et al. 2009], which also estimates the body poses required for training. To initialize the Gaussians, we project the positions and skinning weights of the mesh to the UV domain. After initialization, the mesh is no longer used.

Animation. We drive the animation of the avatar using LBS and non-linear correctives computed from a face code $\mathrm{e_f} \in \mathbb{R}^{32}$, a body code $\mathrm{e_b} \in \mathbb{R}^{32}$. At training time, these codes are estimated from face and body keypoints respectively, which are processed by two separate MLP encoders (see Figure 2). The 2D Gaussian map is then computed as follows:

$$G = G_t + M(D(e_f, e_b)) \tag{4}$$

where $\mathrm{G_t}$ is a learnable template and D is the convolutional decoder that estimates non-linear correctives for the Gaussian positions, rotations, scales and spherical harmonics, in canonical space. And, M is a binary mask that removes Gaussians that align with dead space in the UV domain, reducing the number of Gaussians from 65536 to 60381. We do not apply correctives to the Gaussian δ (i.e. the term used to calculate opacity), since using constant δ greatly simplifies the pruning of non-visible Gaussians. The final step consists in converting the 2D Gaussian map G into 1D arrays of Gaussian properties $\{\mu_c, \Sigma_c, \delta, SH\}$ and applying LBS using the same formulation as in [Li et al. 2024],

$$\begin{aligned} \mu &= \mathrm{R}\mu_c + \mathrm{t} \\ \Sigma &= \mathrm{R}\Sigma_c\mathrm{R}^\top \end{aligned} \tag{5}$$

where R and t are the blended joint rotations and translations, using the skinning weights assigned to each Gaussian at initialization.

Loss Functions. Our loss function is a weighted-sum of several terms: $\mathcal{L}_{photo}$ and $\mathcal{L}_{lpips}$ are the photometric (L1) and LPIPS [Zhang et al. 2018] distance between the ground-truth and predicted image. $\mathcal{L}_{kpt}$ is the L1 loss, and the body pose keypoints. To reduce "run-away" Gaussians that drift far from their initial position, $\mathcal{L}_{offset}$ is an L1 loss that is maximized when the mean μ of each Gaussian in G is unchanged from μ where the Gaussian was initialized at the start of training. To reduce unwanted holes in the avatar, we introduce $\mathcal{L}_\alpha$, which is an L1 loss that is maximized when the α-transparency map generated by the model matches the Sapiens [Khirodkar et al. 2024] segmentation mask; the intuition is that the avatar should be opaque and background pixels should be transparent. When rendering Gaussians on a mobile, it is cheaper if each Gaussian is visible from fewer pixels [Niemeyer et al. 2024]. To reduce the number of pixels to which Gaussian is applied during rendering, we adopt the loss $\mathcal{L}_\sigma$, which uses L1 to minimize the average size of the Gaussians, and we adopt $\mathcal{L}_{opacity}$, which uses L1 to minimize the average opacity of the Gaussians. We visualize the effects of the losses in the supplementary material. The weights for the sum are set to $\lambda_{photo} = 1$, $\lambda_{lpips} = 0.1$, $\lambda_{opacity} = 0.01$, $\lambda_{scale} = 1$, $\lambda_{offset} = 1$, $\lambda_{alpha} = 0.1$, and $\lambda_{kpt} = 0.1$. The loss is computed as

$$\begin{aligned} \mathcal{L} = {} & \lambda_{photo}\mathcal{L}_{photo} + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_\alpha\mathcal{L}_\alpha + \lambda_{kpt}\mathcal{L}_{kpt} \\ & + \lambda_{opacity}\mathcal{L}_{opacity} + \lambda_{scale}\mathcal{L}_{scale} + \lambda_{offset}\mathcal{L}_{offset} \end{aligned} \tag{6}$$

Implementation Details. The face and body MLP encoders consist of 2 hidden layers of size 256 followed by LeakyReLU activations. For better generalization, we apply the inverse head transform to the face keypoints and the inverse root transform to the body keypoints.

The convolutional decoder D begins with a linear layer that projects the input code to $32 \times 4 \times 4$, followed by 6 convolutional blocks that produce a $K \times 256 \times 256$ Gaussian corrective map. Each convolutional block consists of a bilinear upsampling operation followed by two convolutional layers and LeakyReLU activations, and skip connections. The decoder, encoders, and template G_t are trained end-to-end for 300k iterations, which takes around 20h on an NVIDIA A100 GPU.

4.2 Mobile-ready Distillation

Linear Correctives. We now turn our attention distilling the decoder D into linear layers using principal component analysis (PCA). For short, we refer to this process as *linearization.* To distill a decoder for one identity, we begin by collecting a dataset of inputs and outputs to the encoder-decoder. In this section, we continue to use the face-encoder, but we replace the body-encoder with LBS data. So, our input space is a set of poses, and the output space is the output of the decoder from Equation 4.

For one frame, the linearized decoder's input is a pose vector p, which the concatenation of the face encoder's embedding and the body pose in quaternion form. We use PCA to compress the input vector θ to a total of 64 dimensions, which we then augment with a columns of ones to form a design matrix C. We compute the linear layer in the least-squares sense using the normal equations i.e., we form $(C^TC)^{-1}C^T$ as the pseudo-inverse of the design matrix C which is then multiplied by the decoder's output. We describe the linear distillation more precisely in Algorithm 1.

Algorithm 1 Distilled Decoder Correctives Computation

```
Stage 1: Calculate Bases
Input: Set of input poses $\{p_i\}$, Masked Correctives $\{M(D(p_i))\}$
Output: Pose Basis $B_p$, Correctives Basis $B_c$, Mean Pose $\bar{p}$
  1: $\bar{p} \leftarrow \text{mean}(\{p_i\})$
  2: $B_p \leftarrow \text{PCA}(\{p_i - \bar{p}\})$
  3: $C \leftarrow \{[1, (p_i - \bar{p}) \cdot B_p]\}$
  4: $B_c \leftarrow (C^TC)^{-1}C^T \cdot \{M(D(p_i))\}$
  5: return $\bar{p}, B_p, B_c$
Stage 2: Calculate Correctives for New Pose
Input: Pose Basis $B_p$, Correctives Basis $B_c$, Mean Pose $\bar{p}$, Novel Pose $p$
Output: Distilled Decoder Output $D_L(p)$
  1: $c \leftarrow (p - \bar{p}) \cdot B_p$
  2: $D_L(p) \leftarrow [1, c] \cdot B_c$
  3: return $D_L(p)$
```

The distilled decoder is a 2-layer model. The first layer is a linear layer that takes a 64d compressed vector and outputs a 60381x16 vector, and the layer has 64x60381x16 parameters. The second layer takes 6 of the 16 values from the first layer and expands them into 27 spherical harmonics, and the layer has 6x27 parameters. The remaining 10 channels are used for the Gaussian geometric parameters such as rotation, scale, and translation. By far, the dominant cost is the first layer, which we find takes 5 ms to run (with quantization) on the neural processing unit (NPU) of the XR2G2 chip in the Meta Quest 3 VR headset.

With D_L representing the distilled decoder, we compute the Gaussians for one frame of animation as follows.

$$G = LBS(G_t + D_L(p), \theta) \tag{7}$$

Corrective Sharing. While the static components of Gaussians need to be high-frequency, nearby particles of skin, hair, and clothing move together, making the displacements mostly low-frequency. Based on this observation, we propose to reduce the degrees of freedom in the correctives. More specifically, we group nearby Gaussians together and let them share the correctives. This effectively reduces the required rows of a linear matrix, leading to significant memory and computational time reduction on mobile.

While the decoder from Equation 4 produces a size 256x256x37 output, we modify the decoder to produce a smaller 64x64x37 output. For notation, we call the modified decoder D_{GCS}, where GCS is short for Gaussian corrective sharing. This reduces the number of correctives from 65536 to 4096. During training, we take the output of D_{GCS} and use `Nearest` interpolation to upscale it from 64x64 to 256x256 = 65536 correctives.[1] This has the effect of "sharing" correctives for each 4x4 neighborhood of Gaussians in UV-space. For notation, we abbreviate `Nearest` upscaling as Up. During training, we compute the Gaussians for one frame of animation as

$$G = LBS(G_t + M(Up(D_{GCS}(e_f, e_b))), \theta) \tag{8}$$

Finally, to combine linear distillation and corrective sharing, we compute the Gaussians as

$$G = LBS(G_t + M(Up(D_{LGCS}(p))), \theta) \tag{9}$$

We illustrate the end-to-end system with D_{LGCS} and on-device inference in Figure 3.

4.3 Rendering

We have implemented our visualization system in Vulkan to take advantage of the hardware rasterization for accelerated Gaussian splatting, as in [Niedermayr et al. 2024; Xu and Yu 2024]. This is crucial for achieving high performance in compute-constrained setups like mobile devices. We now provide a brief overview of the rendering pipeline in VR. We first run a compute step that animates the avatars with linear blend skinning as proposed in Fig. 2. Next, we perform per-primitive culling to remove Gaussians that are outside the field-of-view for both stereo views in VR. Then, we project and sort the remaining Gaussians according to their view depth and reuse the sorted indices across both eye views. Finally, we expand the projected Gaussians into quad primitives with corresponding colors and opacities, which are rasterized and blended via a traditional graphics pipeline. The rationale for this final step is that Gaussian primitives are ellipsoids, but traditional graphics hardware is optimized for triangles and quadrilaterals. All of the stages are implemented in standard Vulkan compute shaders except for the last (rasterization) stage, which uses a combination of a vertex and a fragment shader.

5 Experimental Setup

We train and evaluate on data that was collected on an internal capture dome with the users' permission to use in published research. Our capture dome is a 3 meter diameter dome with 512, 25 megapixel cameras streaming at 90 FPS. The dome also has 1024 individually controllable lights.

For evaluating the results, we render the avatars in front of an image of the multi-camera, multi-light studio environment where the ground-truth data was captured, and we show examples of

[1]We experimented with both bilinear interpolation and nearest interpolation. See supplementary material for details.

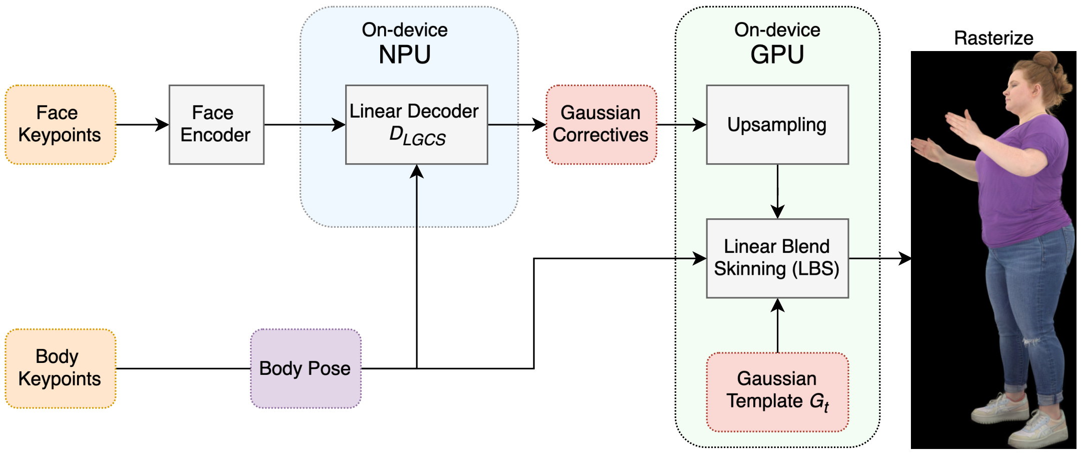


Fig. 3. **System Diagram with optimizations.** We show the end-to-end optimized system, including the techniques from Sections 4.2.

Table 1. **Main results.** We evaluate the impact of linearization and number of correctives on quality. Results are at the original image resolution, with images cropped to rectangles based on segmentation of the avatar. Results are averaged over 4 identities. Latency is for the decoder only. One of the identities in Linear from scratch diverged due to instability and the reported number is the average across the remaining 3 identities.

Model	# Gaussians	# Correctives	L1 ↓	LPIPS ↓	PSNR ↑	SSIM ↑	Decoder Latency on Quest 3
Animatable Gaussians [Li et al. 2024]	300k	300k	0.037	0.143	24.637	0.620	
SqueezeMe (initial)	60k	60k	0.036	0.146	24.992	0.638	
SqueezeMe (GCS)	60k	4k	0.036	0.147	25.024	0.636	
SqueezeMe (Linearized)	60k	60k	0.039	0.150	24.411	0.632	5.0 ms
SqueezeMe (GCS + Linearized)	60k	4k	0.039	0.151	24.370	0.629	0.45 ms
SqueezeMe (no decoder)	60k	0	0.040	0.164	23.905	0.622	0
Linear from scratch	60k	60k	0.044*	0.215*	23.454*	0.616*	5.0 ms

ground-truth and generated images in Figure 4. A significant portion of the image is background, so we crop the rendered images to the maximum width and height of Sapiens [Khirodkar et al. 2024] segmentation masks on the ground-truth data. It is important to note that without this cropping, all models would benefit from artificially inflated accuracy scores due to the large background regions that do not contain the avatar. We use LPIPS [Zhang et al. 2018], L1, PSNR, and SSIM [Wang et al. 2004] to evaluate the cropped images against ground-truth.

Training and evaluation use two separate sets of poses from distinctive animation segments. All results tables, figures, and videos are computed on the evaluation set. Further, in all of our evaluations, we are performing novel-view synthesis, i.e. we are using camera positions that were not in training set. The evaluation set is held-out from the dataset that is used for training, linear distillation, and quantization. For each subject, we use 2000-3000 frames per subject for training and 2000 held-out frames for evaluation. For distillation, we use 512 frames from the training set. All results are averaged across 4 identities and 5 cameras per identity.

For on-device demonstration, we use a Meta Quest 3 VR headset. The Quest 3 is equipped with a Snapdragon XR2 Gen 2 chip, which integrates a CPU, a GPU, and a neural processing unit (NPU) [Qualcomm]. Unlike desktop GPUs, Snapdragon does not support CUDA; however, its mobile GPU is programmable via the Vulkan API, and its NPU can execute quantized neural networks exported from frameworks like PyTorch.

6 Results

On-device Demo. In Figure 1, we illustrate 3 avatars running concurrently on a VR headset. The decoder and Gaussian visualization run locally on the headset. While we focus our evaluation on the Quest 3 headset, many smartphones and VR headsets have similar Qualcomm Snapdragon processors, indicating that similar results are achievable on a broad variety of mobile devices. We provide a VR demo video in the supplementary material.

Quantitative Comparison. In Table 1, we compare several versions of our method with the baseline of Animatable Gaussians (AG) [Li et al. 2024]. AG has 300k Gaussians and can run at 10 FPS on an NVIDIA GPU, but we found it infeasible to run on a mobile device. Without Gaussian corrective sharing (GCS) or linearization, our initial SqueezeMe model has a convolutional decoder with 60k Gaussians and 60k correctives. Quantatively, our initial SqueezeMe model matches the L1 of Animatable Gaussians; it slightly underperforms the LPIPS of AG; and it slightly outperforms the PSNR and SSIM of AG. Thus, our initial model produces comparable visual

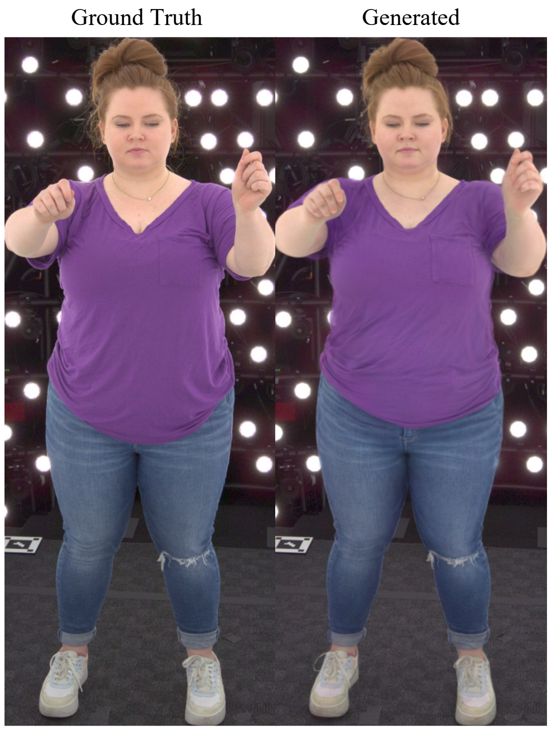


Fig. 4. Qualitative comparison of the mobile-ready avatars. Left: ground-truth image. Right: generated image, with the avatar rendered in front of a static image of the studio. Both images are cropped to a bounding-box of a human segmentation mask.

quality to AG, while having 5x fewer Gaussians. At the other end of the spectrum with lower latency but also lower quality, we propose a version of SqueezeMe that has no decoder and relies entirely on linear blend skinning (LBS) to animate the Gaussian avatar; in Table 1 this model significantly degrades on all metrics compared to AG and the initial SqueezeMe model.

Driving a tradeoff between complexity and quality, we propose a version of SqueezeMe with GCS, which reduces the number of correctives from 60k to 4k and has a smaller convolutional decoder; this model matches the L1 of the initial SqueezeMe model but degrades slightly on LPIPS, PSNR, and SSIM, as seen in Table 1. Finally, we consider SqueezeMe models with linearization; these provide a midpoint between the quality metrics of the initial model and the no-decoder model, with decoder latency as low as 0.45 ms, making the linearized models compelling for practical on-device applications.

To show the effectiveness of the two-stage distillation strategy, let us consider the alternative of training a linear model from scratch without distillation. In particular, we initialize a linear model with random weights and train it with the 7 losses that are used in the main paper. Table 1 shows that the "linear from scratch" model produces results that are significantly worse than the our SqueezeMe model. Note that for one identity, training a linear model from scratch diverged, even with some exploration of learning rate and other hyperparameters. This suggests that training from scratch is less stable than training with distillation. These factors illustrates the benefit of distillation for producing high-quality and low-latency Gaussian avatars.

In the supplementary material, we also evaluate SqueezeMe on the AvatarRex dataset [Zheng et al. 2023b].

Quantization. For on-device inference, we quantized the linear models with 8-bit weights and 16-bit activations. In particular, we used post-training quantization. We found that floating-point and quantized linear models produce identical quantative results. Therefore, L1, LPIPS, PSNR, and SSIM of the "Linearized" results in Table 1 are for both quantized and unquantized models. The latencies reported in the table are for the quantized and linearized models.

Qualitative Comparison. In Figure 5, we show representative examples of the different models across different identities, poses, and camera views. We observe high quality even with significant motion in the arms, torso, and legs. We observe that using corrective-sharing to reduce the number of correctives from 65k to 4k and linearizing the model produce very little degradation in the avatar quality. However, by reducing the number of correctives and linearizing the model, we are able to squeeze the decoder onto a VR headset with just 0.45 ms of latency per inference.

Limitations. We visualize failure-cases in Figure 6. SqueezeMe models intermittently produce blurry hands (see Figure 6(a)), and unwanted transparency (see Figure 6(c)). Further, our optimizations – namely, using Gaussian corrective sharing to reduce the number of correctives from 60k to 4k, and linear distillation – introduce some additional artifacts that appear intermittently during inference. For instance, in Figure 6(b, e) we find that corrective-sharing and linearization can both cause degradation at on the arms, particularly at the armpit and the point where a t-shirt sleeve meets the skin. Finally, in Figure 6(d) we observe that corrective-sharing reduces the visual quality on the seat of the pants, especially when the avatar is standing with legs apart. To offer an intuitive explanation: corrective sharing assumes that neighboring Gaussians move together, but the arm and leg joints Gaussians may move more independently, causing artifacts to sometimes appear in those regions. These problems may be resolved in the future with more adaptive methods of distributing Gaussians and correctives across a human avatar.

7 Conclusion

We have proposed a system comprised of multiple techniques to improve the efficiency of Gaussian Splatting in animatable human avatars, including a compact 3D Gaussian avatar representation, linear distillation, and corrective-sharing. This improves the latency of the Gaussian corrective decoder from a baseline of 50 ms to just 0.45 ms. Further, we show that running 3 avatars at 72 FPS on a Quest 3 VR headset is now possible.

By drastically reducing computational costs, this work not only advances photorealistic avatar rendering in VR but also expands the possibilities for practical applications. These include lifelike telepresence for remote collaboration, enhanced realism in multiplayer VR environments, and greater inclusivity by making high-quality avatars feasible on mobile-grade hardware. Looking ahead, the techniques introduced here could serve as the foundation for scaling avatar systems to support larger numbers of participants and for exploring their integration with augmented reality systems, bridging the gap between physical and virtual worlds.

References

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *2018 International Conference on 3D Vision (3DV)*, pages 98–109. IEEE, 2018a.

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8387–8397, 2018b.

Timur Bagautdinov, Chenglei Wu, Tomas Simon, Fabian Prada, Takaaki Shiratori, Shih-En Wei, Weipeng Xu, Yaser Sheikh, and Jason Saragih. Driving-signal aware full-body avatars. *ACM Transactions on Graphics (TOG)*, 40(4):1–17, 2021.

Renat Bashirov, Alexey Larionov, Evgeniya Ustinova, Mikhail Sidorenko, David Svitov, Ilya Zakharkin, and Victor Lempitsky. Morf: Mobile realistic fullbody avatars from a monocular video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3545–3555, 2024.

Yushuo Chen, Zerong Zheng, Zhe Li, Chao Xu, and Yebin Liu. Meshavatar: Learning high-quality triangular human avatars from multi-view videos. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023a.

Zhaoxi Chen, Fangzhou Hong, Haiyi Mei, Guangcong Wang, Lei Yang, and Ziwei Liu. Primdiffusion: Volumetric primitives diffusion for 3d human generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T Barron. Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

Juergen Gall, Carsten Stoll, Edilson De Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1753. IEEE, 2009.

Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024.

Chen Guo, Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. Vid2avatar: 3d avatar reconstruction from videos in the wild via self-supervised scene decomposition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5875–5884, 2021.

Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. GaussianAvatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Shoukang Hu and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Boyi Jiang, Yang Hong, Hujun Bao, and Juyong Zhang. Selfrecon: Self reconstruction your digital avatar from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5605–5615, 2022a.

Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. Neuman: Neural human radiance field from a single video. In *European Conference on Computer Vision*, 2022b.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42 (4), 2023.

Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Su Zhaoen, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for human vision models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part IV*, page 206–228, Berlin, Heidelberg, 2024. Springer-Verlag.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024.

Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhöfer, Jürgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. In *European Conference on Computer Vision*, pages 419–436. Springer, 2022.

Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable Gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19711–19722, 2024.

Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. High-fidelity and real-time novel view synthesis for dynamic scenes. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–9, 2023.

Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor. *ACM Transactions on Graphics (TOG)*, 40:1 – 16, 2021.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2015a.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015b.

Shugao Ma, Tomas Simon, Jason Saragih, Dawei Wang, Yuecheng Li, Fernando De La Torre, and Yaser Sheikh. Pixel codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 64–73, 2021.

Shengjie Ma, Yanlin Weng, Tianjia Shao, and Kun Zhou. 3d gaussian blendshapes for head avatar animation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–10, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10349–10358, 2024.

Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv:2403.13806*, 2024.

Haokai Pang, Heming Zhu, Adam Kortylewski, Christian Theobalt, and Marc Habermann. Ash: Animatable gaussian splats for efficient and photoreal human rendering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Animatable neural radiance fields for human body modeling. In *International Conference on Computer Vision (ICCV)*, 2021a.

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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.

Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic point fields. *arXiv preprint arXiv:2304.02626*, 2023.

Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Qualcomm. Meta quest 3. https://www.qualcomm.com/products/mobile/snapdragon/xr-vr-ar/xr-vr-ar-device-finder/meta-quest-3.

Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023.

Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

Edoardo Remelli, Timur Bagautdinov, Shunsuke Saito, Chenglei Wu, Tomas Simon, Shih-En Wei, Kaiwen Guo, Zhe Cao, Fabian Prada, Jason Saragih, et al. Drivable volumetric avatars using texel-aligned features. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022.

Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J Black. Scanimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2886–2897, 2021.

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference* *on Computer Vision and Pattern Recognition*, pages 1606–1616, 2024.

Shih-Yang Su, Frank Yu, Michael Zollhoefer, and Helge Rhodin. A-neRF: Articulated neural radiance fields for learning human shape, appearance, and pose. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Shih-Yang Su, Timur M. Bagautdinov, and Helge Rhodin. Npc: Neural point characters from video. *ArXiv*, abs/2304.02013, 2023.

Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20675–20685, 2024.

David Svitov, Pietro Morerio, Lourdes Agapito, and Alessio Del Bue. Haha: Highly articulated gaussian human avatars with textured mesh prior. *ACCV*, 2024.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)*, 38(4): 1–12, 2019.

Liao Wang, Kaixin Yao, Chengcheng Guo, Zhirui Zhang, Qiang Hu, Jingyi Yu, Lan Xu, and Minye Wu. Videorf: Rendering dynamic radiance fields as 2d feature video streams. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 470–481, 2024a.

Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. Vˆ 3: Viewing volumetric videos on mobiles via streamable 2d dynamic gaussians. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024b.

Shaofei Wang, Katja Schwarz, Andreas Geiger, and Siyu Tang. Arah: Animatable volume rendering of articulated human sdfs. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, page 1–19, Berlin, Heidelberg, 2022. Springer-Verlag.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Chung-Yi Weng, Brian Curless, and Ira Kemelmacher-Shlizerman. Vid2actor: Free-viewpoint animatable person synthesis from video in the wild. *arXiv preprint arXiv:2012.12884*, 2020.

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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021.

Minye Wu, Zehao Wang, Georgios Kouros, and Tinne Tuytelaars. Tetrirf: Temporal tri-plane radiance fields for efficient free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6487–6496, 2024.

Zhen Xu and Zhiyuan Yu. Fast gaussian rasterization. https://github.com/dendenxu/fast-gaussian-rasterization, 2024.

Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20029–20040, 2024a.

Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Representing long volumetric video with temporal gaussian hierarchy. *ACM Transactions on Graphics*, 43(6), 2024b.

Peizhi Yan, Rabab Ward, Qiang Tang, and Shan Du. Gaussian deja-vu: Creating controllable 3d gaussian head-avatars with enhanced generalization and personalization abilities. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 276–286, 2025.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024.

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21057–21067, 2023a.

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, 2023b.

Heming Zhu, Fangneng Zhan, Christian Theobalt, and Marc Habermann. Trihuman: A real-time and controllable tri-plane representation for detailed human geometry and appearance synthesis. 44(1), 2024.

Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. *arXiv.org*, 2311.08581, 2023.

Wojciech Zielonka, Timo Bolkart, Thabo Beeler, and Justus Thies. Gaussian Eigen Models for Human Heads. *arXiv:2407.04545*, 2024.

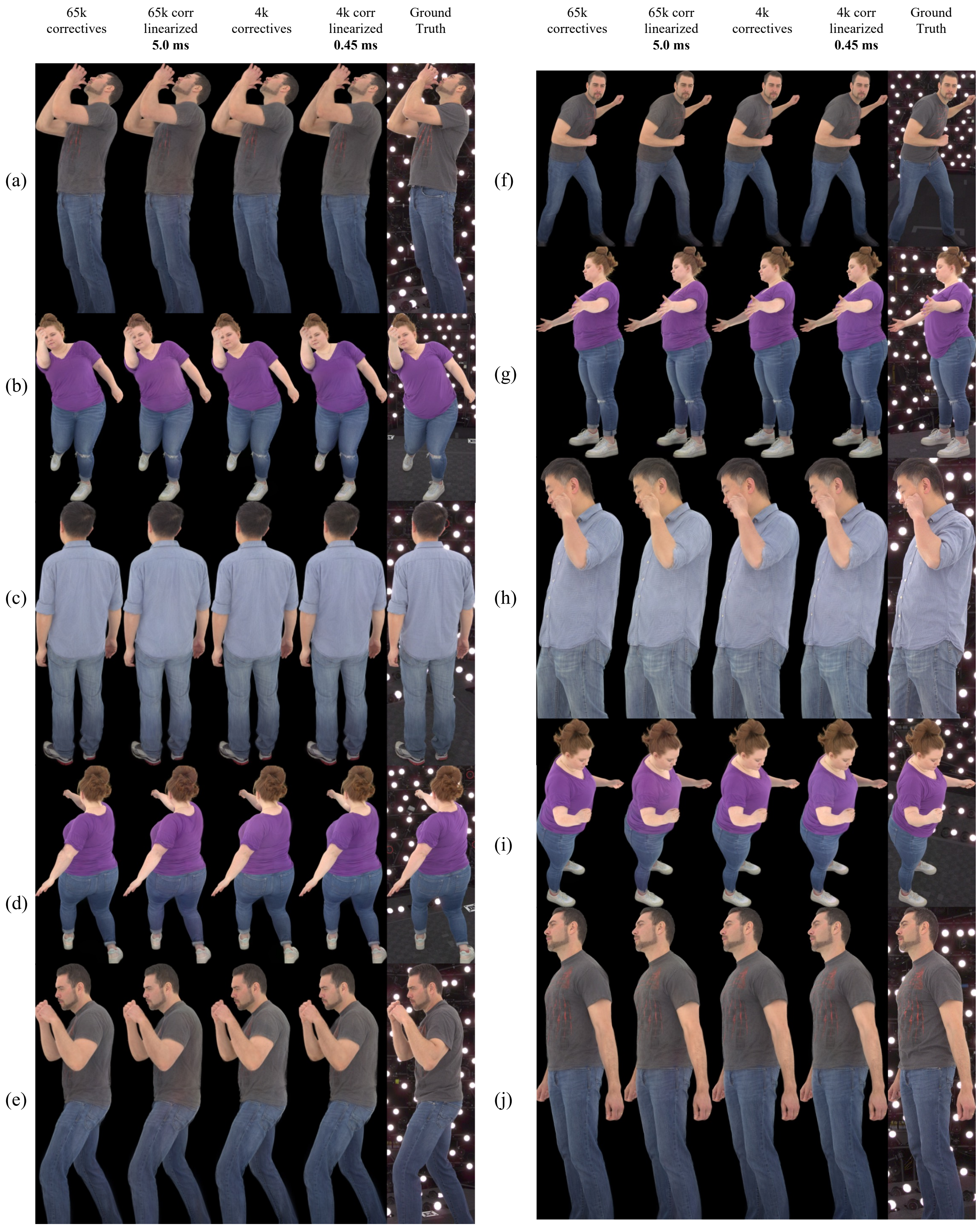
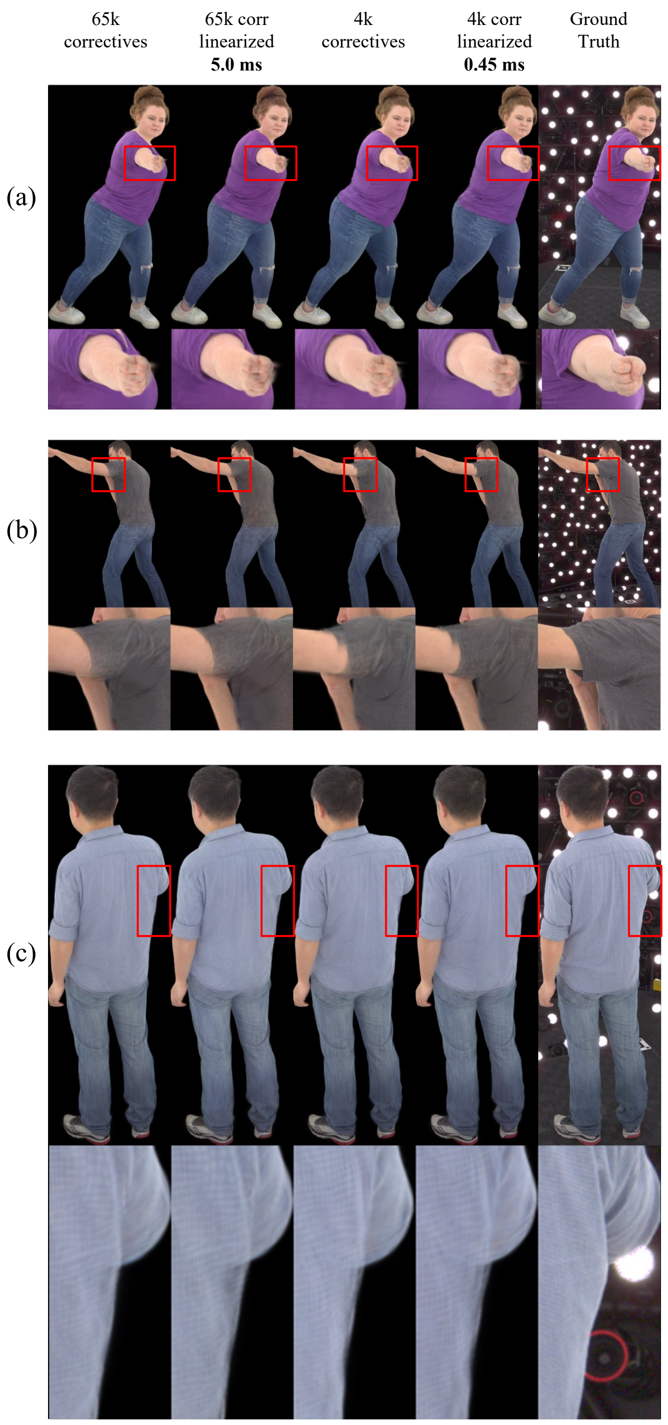


Fig. 5. **Qualitative results.** Our 0.45 ms model produces results that are competitive with far more expensive models.



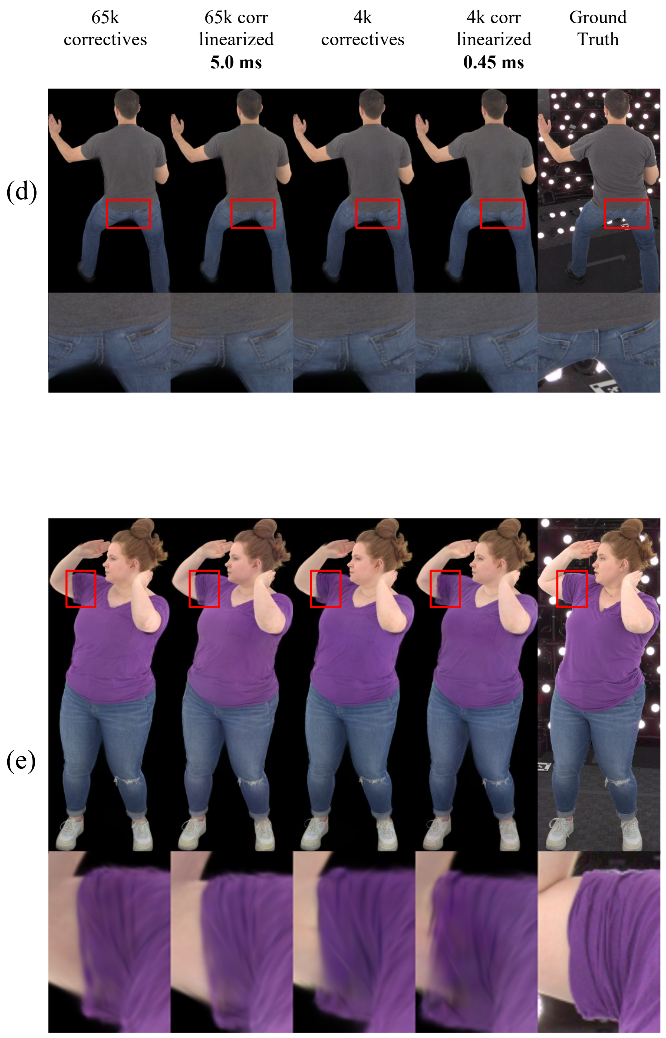


Fig. 6. **Failure cases.** (a) In all models and identities, hands are sometimes blurry. (b) The 4k and linearized models struggle with the edge of a t-shirt sleeve. (c) All models have unwanted transparency under the arm for this identity's avatar, but it is worse in 4k and linearized models. (d) All models struggle with the seat of the pants for this identity, but the 4k and linearized models struggle more. (e) The 4k and linear models suffer more degradation in the underarm for this identity.

Appendix

A Upsampling Method

Our initial SqueezeMe decoder has 65k outputs, which are reduced to 60k correctives by masking. These 60k pose-dependent correctives are applied to the 60k Gaussians to improve their appearance. With Gaussian corrective sharing (GCS), the decoder has 4k outputs, which we upsample to 65k and then mask to 60k. In this section, we compare two methods for upsampling: Nearest and Bilinear interpolation. We present the comparison in Table 2. The numerical results show that Bilinear has a small advantage in quality over Nearest upsampling.

Further, we provide a qualitative evaluation in Figure 7 and find that the two methods produce very similar results. For ease of implementation, and to reduce the load on the mobile GPU, we selected nearest interpolation for our VR demo videos and for the GCS results in the main paper.

B Further Discussion of Loss Functions

In the main paper, we trained with 7 loss functions: $\mathcal{L}_{photo}$, $\mathcal{L}_{lpips}$, $\mathcal{L}_{\alpha}$, $\mathcal{L}_{kpt}$, $\mathcal{L}_{opacity}$, $\mathcal{L}_{scale}$, and $\mathcal{L}_{offset}$. In Figure 8, we illustrate the impacts of these losses.

C Results on AvatarRex Dataset

So far, this paper and supplementary material have focused on an internal dataset that was captured with 512 cameras. Now, we consider the AvatarRex dataset, which was proposed in [Zheng et al. 2023b].[2] The AvatarRex dataset has 4 identities and 16 cameras, with approximately 2000 frames per identity. In particular, 14 camera views are used for training and 2 cameras are held-out for evaluation. Further, for each avatar we hold out 500 frames per evaluation, therefore the training poses and evaluation poses are different. Note that prior works such as [Li et al. 2024] use the same poses for training and evaluation. For all our experiments on this dataset, we report results averaged across the identities zzr and lbn2.

We present results on AvatarRex in Table 3. When comparing Animatable Gaussians (AG) with 300k Gaussians to SqueezeMe with 60k Gaussians and 60k correctives, we observe that SqueezeMe outperforms AG on L1, PSNR, and SSIM. Further, when comparing AG to comparing SqueezeMe with GCS and just 4k correctives, we find that SqueezeMe (GCS) outperforms AG on L1 and SSIM, and SqueezeMe (GCS) matches AG on L1. Meanwhile, AG outperforms SqueezeMe on LPIPS. Finally, we visually compare SqueezeMe and AG in Figure 9, and we observe similar quality for both methods.

[2]Specifically, we use the AvatarRex data that is published at https://github.com/lizhe00/AnimatableGaussians/blob/master/AVATARREX_DATASET.md.

Table 2. **Comparison of upsampling methods for Gaussian corrective sharing.** Results are averaged over 4 identities and are directly comparable to the Main Results table in the paper.

Model	Upsampling	# Gaussians	# Correctives	L1 ↓	LPIPS ↓	PSNR ↑	SSIM ↑
SqueezeMe (GCS)	Nearest	60k	4k	0.036	0.147	25.024	0.636
SqueezeMe (GCS + Linearized)	Nearest	60k	4k	0.039	0.151	24.370	0.629
SqueezeMe (GCS)	Bilinear	60k	4k	0.036	0.146	25.075	0.638
SqueezeMe (GCS + Linearized)	Bilinear	60k	4k	0.039	0.150	24.399	0.630

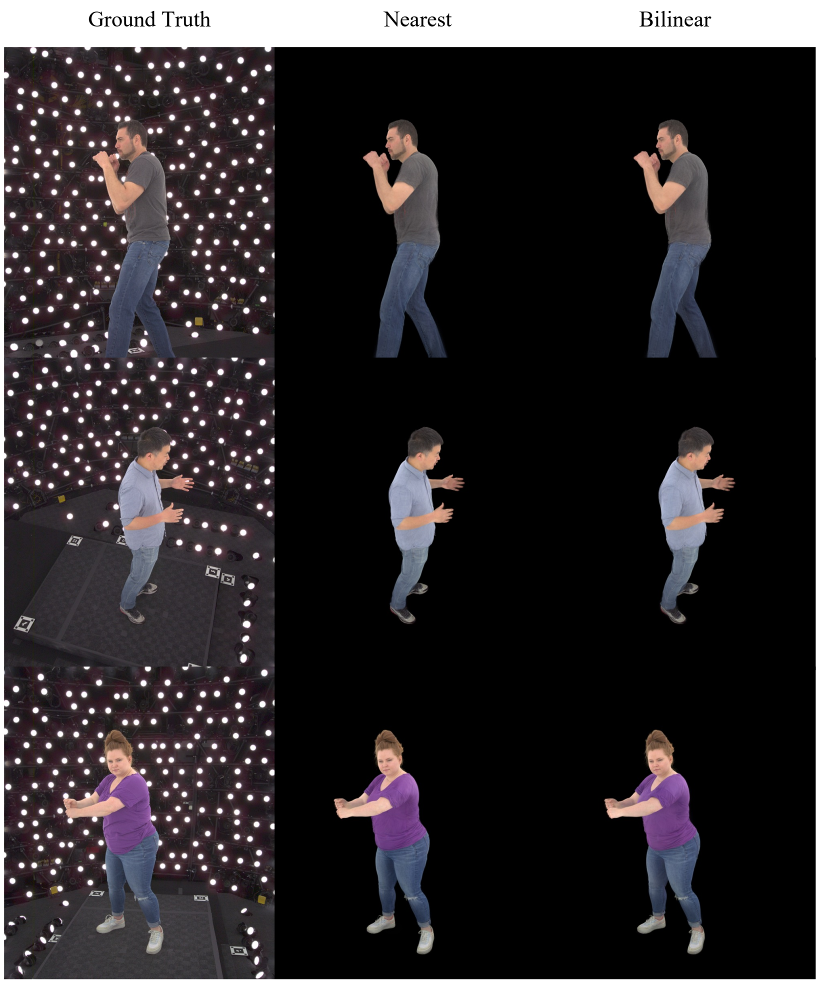


Fig. 7. **Comparing Bilinear and Nearest interpolation in SqueezeMe (GCS).**

Table 3. **Results on AvatarRex.** Results are averaged over two identities.

Model	# Gaussians	# Correctives	L1 ↓	LPIPS ↓	PSNR ↑	SSIM ↑
Animatable Gaussians [Li et al. 2024]	300k	300k	0.059	0.151	19.542	0.844
SqueezeMe	60k	60k	0.057	0.156	20.178	0.851
SqueezeMe (GCS)	60k	4k	0.059	0.158	20.051	0.849

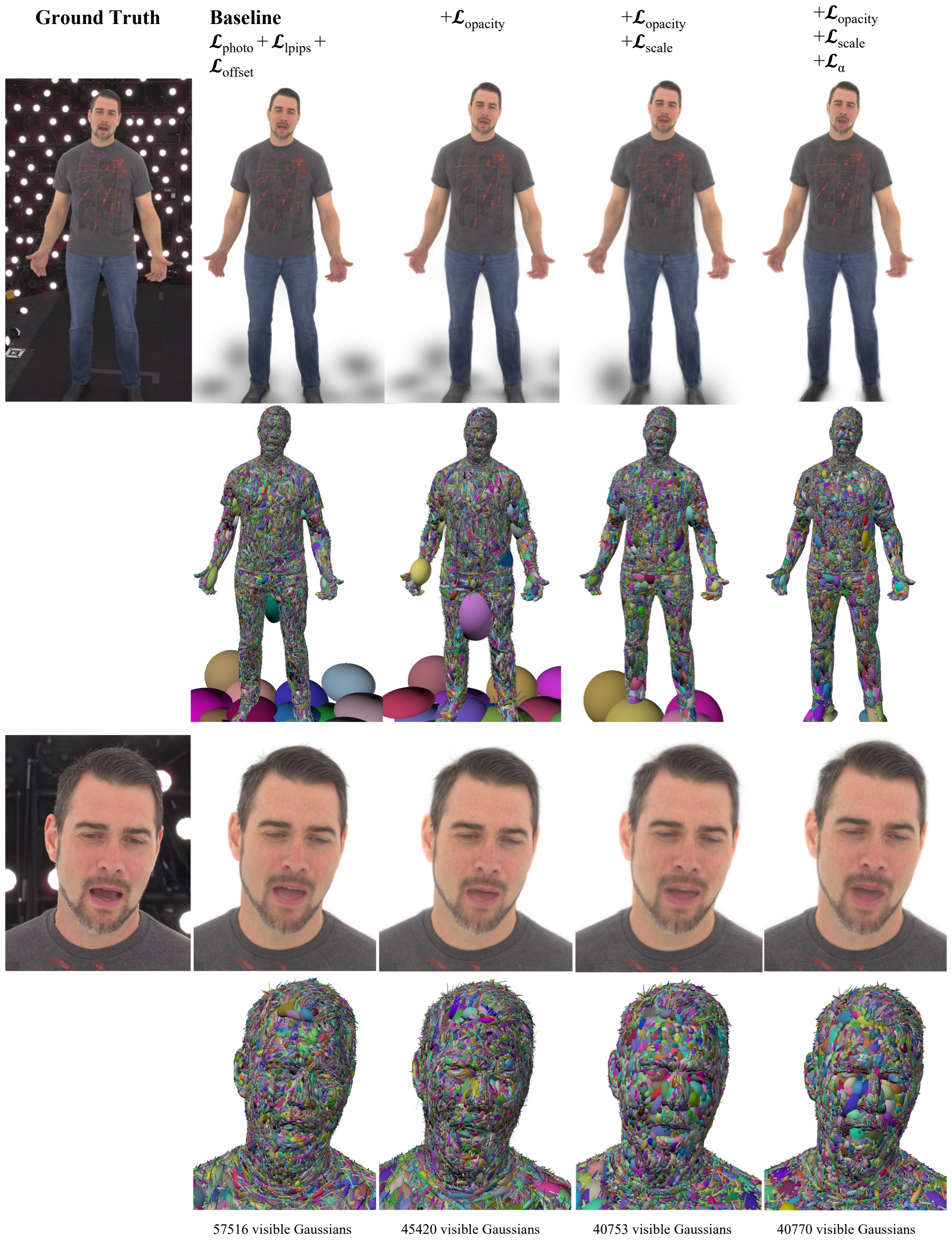

Ground Truth
Baseline
$\mathcal{L}_{\text{photo}} + \mathcal{L}_{\text{lpips}} + \mathcal{L}_{\text{offset}}$
$+\mathcal{L}_{\text{opacity}}$
$+\mathcal{L}_{\text{opacity}}$ $+\mathcal{L}_{\text{scale}}$
$+\mathcal{L}_{\text{opacity}}$ $+\mathcal{L}_{\text{scale}}$ $+\mathcal{L}_{\alpha}$
57516 visible Gaussians
45420 visible Gaussians
40753 visible Gaussians
40770 visible Gaussians

Fig. 8. **Qualitative ablation study.** This figure evaluates the effect of the losses used to train the convolutional decoder. Using the losses of Animatable Gaussians [Li et al. 2024] as our baseline, we incorporate an opacity and scale loss to encourage the model to use less and smaller Gaussians. Prunning the non-visible Gaussians reduces the Gaussian count of the full-body avatar by 28% with minimal quality loss. The alpha loss mitigates transparency issues and removes floaters around the avatar, greatly improving the final result.

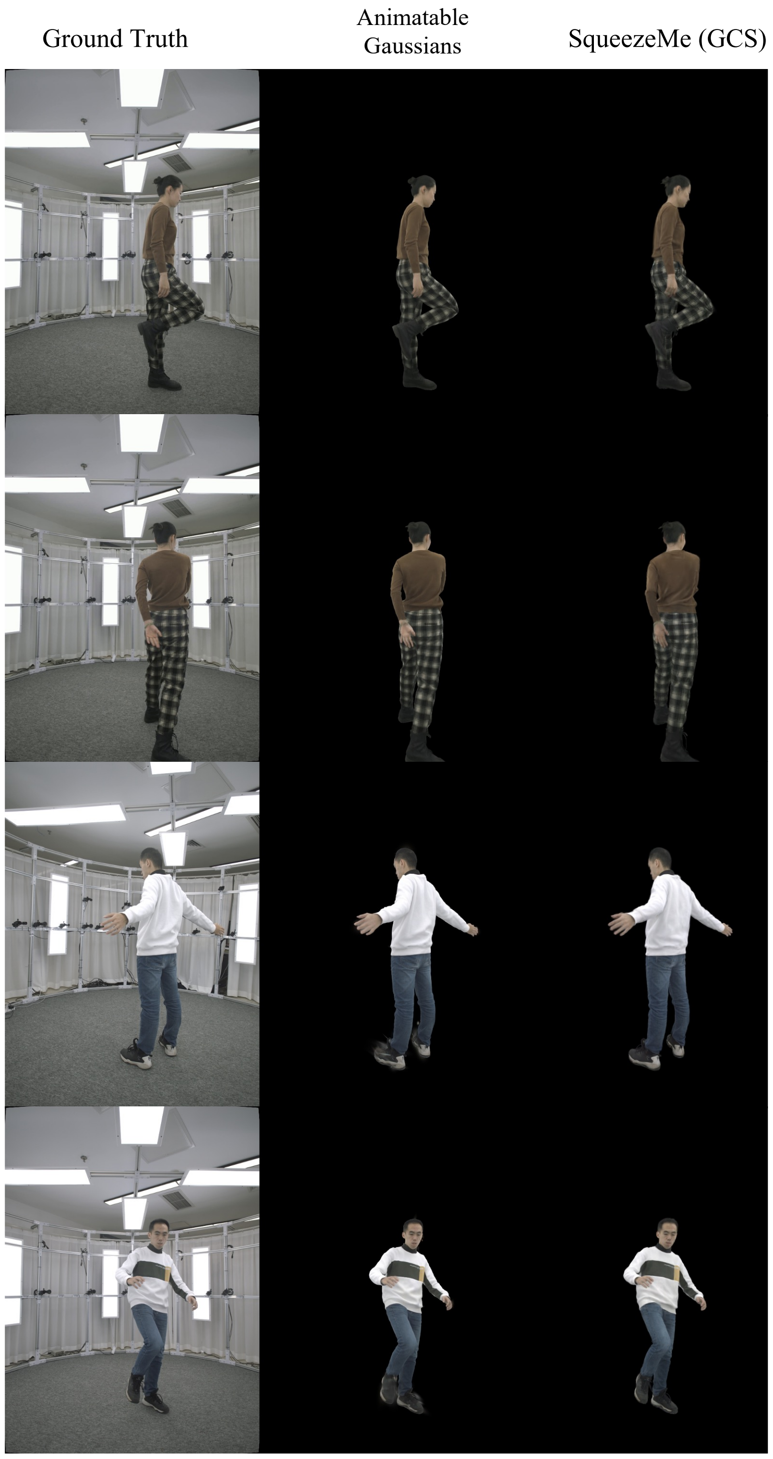


Fig. 9. **Qualitative results on AvatarRex.**