LUCAS: Layered Universal Codec Avatars Supplementary Material

Di Liu[1,2] Teng Deng[1] Giljoo Nam[1] Yu Rong[1] Stanislav Pidhorskyi[1] Junxuan Li[1]
Jason Saragih[1] Dimitris N. Metaxas[2] Chen Cao[1]
[1]Codec Avatars Lab, Meta [2]Rutgers University

1. Network Architecture

In this section, we provide more details of our network architecture and hyperparameters for our id-conditioned hypernetwork, appearance deocder, geometry decoder, Gaussian hypernetwork and Gaussian decoder, respectively.

Identity-conditioned hypernetwork. We adopt a U-Net [2] architecture as our identity-conditioned hypernetwork that takes as input the neutral geometry and texture of a subject to predict subject-specific decoder parameters. The network consists of two parallel downsampling branches that process geometry and texture features separately, followed by a joint upsampling branch. The input geometry and texture are first normalized by subtracting their respective means and dividing by their standard deviations. The geometry branch processes the normalized geometry (scaled by 0.2) while the texture branch processes the normalized texture (scaled by 0.4) through a series of downsampling blocks. The network's channel dimensions progressively increase through the layers with sizes of (3, 32, 64, 128, 256, 256, 512, 512, 512, 512, 256). Features from both branches are concatenated at each scale and processed through the upsampling branch. The network generates three types of outputs through Transfer modules (Fig. 1(a)): untied bias parameters (Θ_{id}), per-identity geometry displacement (d), and per-identity positional encoding (f). Each Transfer module consists of two weight-normalized convolution (Conv2DWN) layers with learnable biases, mapping the concatenated features through a hidden dimension of 512 channels, followed by LeakyReLU activation ($\alpha = 0.2$), before outputting the final parameters. For both face and hair branches, six texture bias parameters are generated for the appearance decoder, corresponding to different resolution levels. The geometry decoder receives five geometry displacement parameters for its Block modules, with an additional displacement map (d) for the output layer. The pixel decoder receives a positional encoding (f) of dimension [4, 1024, 1024] generated by processing the concatenation of the final upsampled features and the input texture through a Transfer module. The architecture leverages weight normalization throughout its convolution and linear layers for stable training, with careful initialization using Glorot initialization scaled by 0.2 for most layers and 1.0 for the final convolution layers in the Transfer modules.

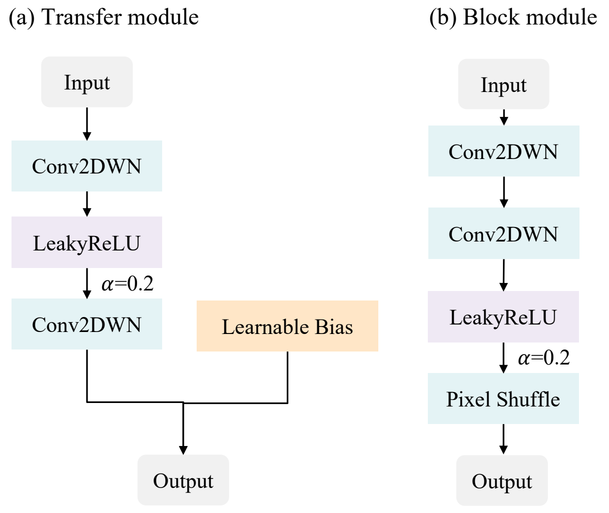


Figure 1. **Architecture of (a) Transfer module and (b) Block module.** (a) Transfer Block processes features through two Conv2dWN layers with intermediate LeakyReLU activation and scaled learnable bias. (b) Block module combines Conv2dWN, LeakyReLU activation, and pixel shuffle operations for feature transformation and upsampling.

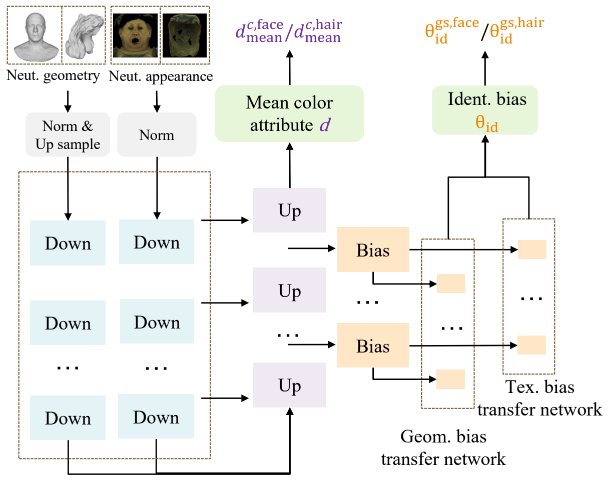
Appearance decoder. The face appearance decoder $\mathcal{D}_e^{\text{face}}$ processes expression encoding z, view-dependent conditions ω and neck pose η to generate detailed textures. Specifically, $z \in \mathbb{R}^{16}$ is first processed by a Block module Fig. 1(b)) to obtain features in $\mathbb{R}^{128}$, while the view direction encoding $\omega \in \mathbb{R}^3$ and neck pose $\eta \in \mathbb{R}^6$ are transformed through a linear layer and LeakyReLU activation to $\mathbb{R}^{16}$. These combined features are then processed through a cascade of Block modules that progressively upsample the spatial resolution from 8×8 to 256×256 while reducing the channel dimensions $(160, 64, 32, 16, 12, 8, 4)$.

The view-dependent conditioning enables the network to capture view-dependent effects like specular highlights and shading variations. For hair appearance decoder $\mathcal{D}_e^{\text{hair}}$, we use a similar network architecture with additional head pose h encoded to $\mathbb{R}^{16}$ through linear layers and LeakyReLU activations, resulting in a 176-dimensional feature vector after concatenation.

Geometry decoder. The face geometry decoder $\mathcal{D}_g^{\text{face}}$ combines expression encoding z with neck pose parameters η to capture expression-dependent geometry deformations and neck movements. The input $z \in \mathbb{R}^{16}$ is first processed by a Block module, expanding the features to $\mathbb{R}^{128}$. Meanwhile, the neck pose η is encoded through a linear layer and LeakyReLU to $\mathbb{R}^{16}$. These concatenated features are processed through the same progressive upsampling architecture as the appearance decoder. Notably, the geometry decoder includes a mask output that helps handle occlusions or invalid regions that may occur due to extreme neck poses or expressions. For hair geometry decoder $\mathcal{D}_g^{\text{hair}}$, the network additionally conditions on head pose h, which is encoded similarly to η, resulting in a 160-dimensional feature vector after concatenation.

Gaussian-splatting hypernetwork. As shown in Fig. 2, our Gaussian hypernetwork $\mathcal{E}_{\text{id}}^{\text{gs}}$ shares the same U-Net backbone as the PiCA hypernetwork, consisting of two parallel downsampling branches that process geometry $\mathbf{G}_{\text{neu}}$ and texture $\mathbf{T}_{\text{neu}}$ features separately. Through the dedicated downsampling branches, the input geometry and texture are first normalized ($0.2\times$ and $0.4\times$ respectively) and processed into multi-scale feature maps. The resulting features are concatenated at each scale and processed through an upsampling network. Through a sequence of transfer modules, the network predicts the identity-specific bias map $\Theta_{\text{id}}^{\text{gs}}$ and extracts mean color attributes d_{mean}^{c} from the neutral appearance data, as formulated in Eq. 7 of the main paper. These bias parameters control the position, rotation, scale, opacity and color attribute of the splatted Gaussians anchored at mesh vertices $\{\hat{t}_k\}_{k=1}^{M}$, allowing for person-specific rendering characteristics. The same architecture is employed for both face and hair branches, enabling joint optimization of all rendering components through a unified hypernetwork backbone.

Gaussian decoder. The Gaussian decoder $\mathcal{D}^{\text{gs}}$ consists of a progressive upsampling network that transforms input features to Gaussian attributes. Taking as input a 128-dimensional expression encoding concatenated with a 16-dimensional neck pose encoding η, the network first processes them through two MLP layers to obtain 8×8 feature maps. These features are then gradually upsampled through a series of transposed convolution layers with LeakyReLU activations ($\alpha = 0.2$), expanding the spatial resolution from 8×8 to 1024×1024 while progressively reducing the channel dimensions $(256, 128, 64, 32, 16, 59)$. The final layer outputs a 59-channel feature map, where the first 49 channels encode the spherical harmonics coefficients for appearance, and the remaining 10 channels encode the Gaussian attributes: position delta δt_k, rotation quaternion q_k, and scale s_k. The quaternions are normalized and scales are constrained through a softplus activation multiplied by 0.5. An additional sigmoid activation is applied to the first spherical harmonic coefficient to obtain the opacity o_k. The face and hair branches share the same architecture but operate independently to handle their respective geometry and appearance characteristics.



Neut. geometry
Neut. appearance
Norm & Up sample
Norm
$d_{\text{mean}}^{c,\text{face}} / d_{\text{mean}}^{c,\text{hair}}$
Mean color attribute d
$\theta_{\text{id}}^{\text{gs,face}} / \theta_{\text{id}}^{\text{gs,hair}}$
Ident. bias θ_{id}
Down
Up
Bias
Tex. bias transfer network
Geom. bias transfer network

Figure 2. **Overview of Gaussian hypernetwork.** The network processes normalized geometry $\mathbf{G}_{\text{neu}}$ and texture $\mathbf{T}_{\text{neu}}$ through parallel branches to predict identity-specific bias parameters $\Theta_{\text{id}}^{\text{gs}}$ and mean color attributes d_{mean}^{c} for Gaussian rendering. The same architecture is used for both face and hair branches.

Gaussian Rendering. After obtaining the Gaussian attributes from both face and hair decoders, we concatenate their features for joint rendering. Let $N = N_f + N_h$ denote the total number of Gaussians, where N_f and N_h represent face and hair Gaussians respectively. We combine their position deltas $\delta t \in \mathbb{R}^{N\times 3}$, rotation quaternions $q \in \mathbb{R}^{N\times 4}$, scales $s \in \mathbb{R}^{N\times 3}$, opacities $o \in \mathbb{R}^{N\times 1}$, and spherical harmonics coefficients $d^c \in \mathbb{R}^{N\times 48}$. For RGB rendering, the color attribute $c_k \in \mathbb{R}^{48}$ for the k-th Gaussian is computed as:

$$c_k = \begin{bmatrix} d_k^{c,\text{base}} + d_{\text{mean}}^{c} \\ \beta \cdot d_k^{c,\text{ho}} \end{bmatrix} \tag{1}$$

where $d_k^{c,\text{base}} \in \mathbb{R}^3$ is the base color component (first three coefficients), $d_{\text{mean}}^{c} \in \mathbb{R}^3$ is the mean color vector, $d_k^{c,\text{ho}} \in \mathbb{R}^{45}$ represents the higher-order coefficients (remaining 45 coefficients). β is a scaling factor for the higher-order terms where we set as 0.05 in our experiment.

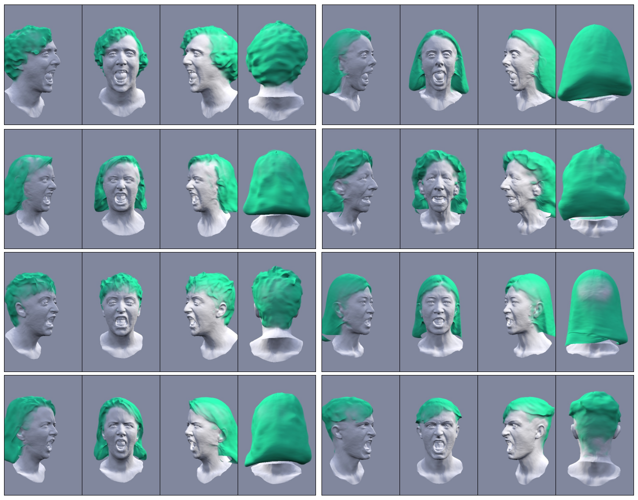


Figure 3. **More visualization of avatar dehairing.** Our method successfully removes hair while preserving the underlying head geometry across subjects with diverse hairstyles.

2. Training details

Our model follows a two-phase training process. In the first phase, we train only the PiCA branch using reconstruction loss $\mathcal{L}_{\text{pica}}$ and dehairing loss $\mathcal{L}_{\text{dehair}}$. Subsequently, we freeze these weights and train the Gaussian branch with loss $\mathcal{L}_{\text{gs}}$. The loss weights are configured as follows:

- Gaussian loss $\mathcal{L}_{\text{gs}}$: $\lambda_{\text{render}} = 10.0$, $\lambda_{\text{scale}} = 0.2$, $\lambda_{\Delta} = 0.01$;
- PiCA loss $\mathcal{L}_{\text{pica}}$: $\lambda_I = 4.0$, $\lambda_D = 10.0$, $\lambda_N = 1.0$, $\lambda_M = 0.1$, $\lambda_S = 4.0$, $\lambda_{KL} = 0.001$, $\lambda_{seg} = 2.0$;
- Dehairing loss $\mathcal{L}_{\text{dehair}}$: initially $\lambda_{\text{dehair}} = 20.0$, decaying to 0 between 70k and 80k iterations;
- $\lambda_{\text{pica}} = \lambda_{\text{gs}} = \lambda_{\text{dehair}} = 1.0$.

We adopt L1 loss for both $\mathcal{L}_I$ and $\mathcal{L}_{\text{render}}$ due to its effectiveness in preserving fine hair details. Hair segmentation regularization $\mathcal{L}_{\text{seg}}$ is restricted to the first phase to avoid Gaussian blur artifacts between hair strands. The high initial weight of dehairing loss $\mathcal{L}_{\text{dehair}}$ accelerates the convergence of bald geometry, ensuring accurate dehaired results without interference from the hair mesh. The model is trained using Adam optimizer with a learning rate of 0.001 for 300k (PiCA branch) + 300k (GS branch) iterations on 4 NVIDIA A100 GPUs.

3. Discussion

Avatar dehairing. We provide more visualization of the dehairing results in Fig. 3. The examples demonstrate that our method can effectively remove diverse hairstyles while maintaining accurate head shape and facial features.

Impact of training strategy. We investigate two different training strategies for our Gaussian model: two-stage training and joint training. In the two-stage approach, we first train the mesh model with $\mathcal{L}_{\text{pica}}$ and $\mathcal{L}_{\text{dehair}}$ for 300k iterations, then freeze the mesh branch and train the Gaussian branch with $\mathcal{L}_{\text{pica}}$ for another 300k iterations. In the joint training approach, we train both branches simultaneously with all losses enabled. As shown in Table 1, both strategies achieve comparable performance across all metrics, with differences being negligible (PSNR: ±0.003, SSIM: ±0.0005, LPIPS: ±0.0007). However, we observe that joint training exhibits significant instability during the initial 10k iterations, often leading to training explosions that substantially delay the convergence process. Given these findings, we adopt the two-stage training strategy in our final model for its superior training stability while maintaining equivalent performance.

Table 1. **Ablation study** on training strategy. While both strategies achieve similar final performance, two-stage training offers better stability during the optimization process.

Strategy	PSNR ↑	SSIM ↑	LPIPS ↓
Two-stage training	**34.5607**	0.9196	**0.2387**
Joint training	34.5579	**0.9201**	0.2394

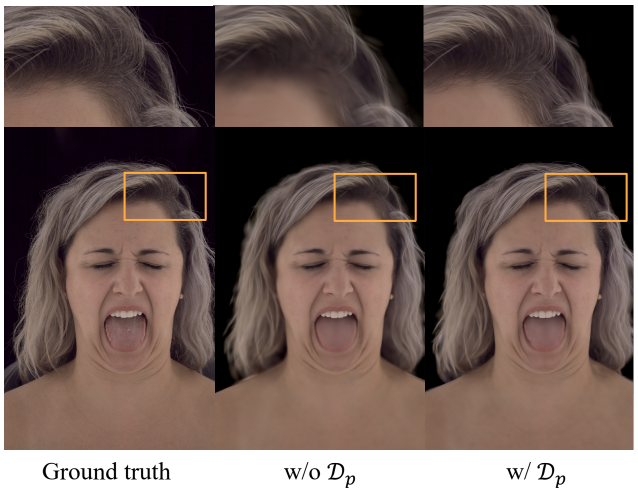


Figure 4. **Impact of pixel decoder $\mathcal{D}_p$.** The pixel decoder helps achieve better facial details and overall fidelity.

Impact of pixel decoder. To understand the role of the pixel decoder in our Gaussian model, we conduct experiments by removing $\mathcal{D}_p$ and supervising the reconstruction solely through Gaussian rendering loss. Note that this ablation is conducted with joint training of the mesh and Gaussian branches. As shown in Fig. 4, models with $\mathcal{D}_p$ achieve noticeably better visual fidelity compared to those without. The quantitative results in Table. 2 further support this observation, with our full model outperforming the variant without $\mathcal{D}_p$ across most metrics, particularly on unseen subjects. We attribute this improvement to the pixel decoder's

Table 2. **Ablation study** on pixel decoder $\mathcal{D}_p$. We evaluate our universal model on both training and unseen subjects. The top two techniques are highlighted in red and yellow, respectively.

	Training subjects			Unseen subjects		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
w/o $\mathcal{D}_p$	34.2113	0.9164	0.2390	32.1328	0.8993	0.2607
URAvatar	33.1209	0.9021	0.2493	31.4462	0.8922	0.2625
Full model	34.5579	0.9201	0.2394	32.5847	0.9087	0.2496

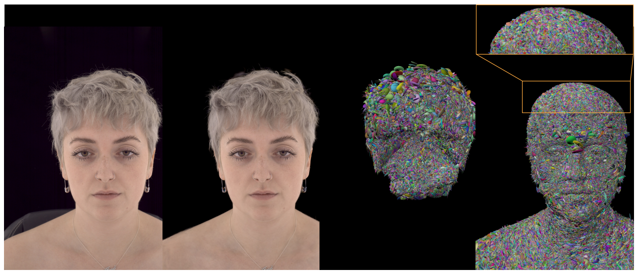


Figure 5. **Impact of Gaussian position prior regularization.** The position delta loss $\mathcal{L}_\Delta$ effectively constrains the Gaussians to stay within their respective regions, leading to clean boundaries between hair and bald head areas and faithful reproduction of the ground truth appearance.

ability to enhance the underlying avatar geometry, which in turn provides better spatial anchoring for Gaussian rendering. Notably, the performance gap widens on unseen subjects, suggesting that $\mathcal{D}_p$ contributes to better generalization of our model. Furthermore, even without the pixel decoder, our model significantly outperforms URAvatar [1], demonstrating the inherent advantage of our layered representation design.

Impact of Gaussian position prior regularization. In Fig. 5 we show the Gaussian shapes for the hair and bald head regions. With the position delta loss $\mathcal{L}_\Delta$, our model effectively maintains the Gaussian primitives in their designated regions - hair Gaussians properly represent the hair volume while face Gaussians accurately cover the bald head area. As shown in the visualization, the Gaussian distributions align well with the ground truth appearance, leading to high-quality rendered results. The clear separation between hair and bald head Gaussians demonstrates the effectiveness of our position regularization.

Impact of training data. To understand how the number of training identities impacts our model's performance, we conducted experiments with varying numbers of subjects in the training set. Specifically, we tested our model using seven different training scales: 4, 8, 16, 32, 48, 64, and 76 identities. The model's performance shows consistent enhancement as we increase the training set size. This trend highlights the importance of diverse training samples in building robust prior representations.

Table 3. **Inference time comparison** of different avatar reconstruction methods. All times measured on NVIDIA A100.

Method	Inference Time (ms)	
	1024^2	512^2
URAvatar [1]	12.84	6.40
LUCAS	12.01	6.42

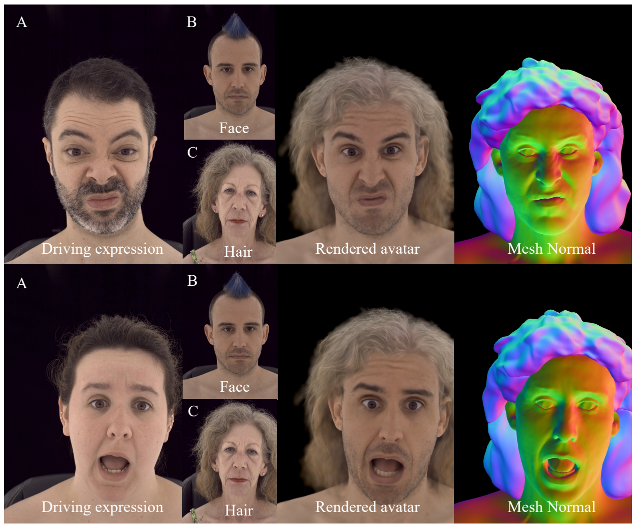


Figure 6. **Application on hairstyle switching.** We combine face condition from subject B, hair condition from subject C, and expressions from subject A. Note how our model maintains high-fidelity facial details while accurately preserving the characteristics of both the chosen face and hairstyle.

Rendering Performance. For all identities, we use a layered structure with separate Gaussian representations for face and hair. We experiment with two configurations: a high-quality setting using $M = 1024 \times 1024 = 1$ Mi Gaussians total (0.5 Mi each for face and hair), and a faster setting with $M = 512 \times 512 = 0.25$ Mi Gaussians total. We observe that increasing the number of Gaussians leads to quality improvement at the cost of slower rendering. As shown in Tab. 3, our complete model with 1 Mi Gaussians takes 12.01 ms for rendering, while reducing to 0.25 Mi Gaussians achieves faster rendering at 6.42 ms on NVIDIA A100. Our method achieves comparable rendering speed to other Gaussian Splatting-based approaches.

4. Application

We can independently control the hair and face appearance by using condition data from different identities. In Fig. 6, we demonstrate this capability by combining the face from

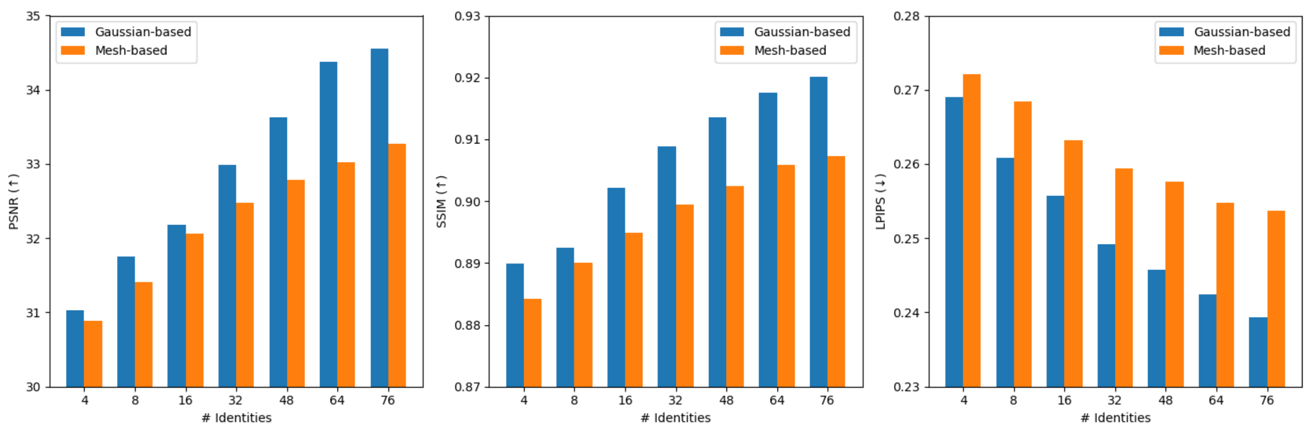


Figure 7. **Ablation study** on the number of training subjects. The model's performance improves consistently with larger training sets, demonstrating the importance of diverse subjects for learning robust priors.

one subject, hair from another, and driving expressions using a third subject. Our model successfully preserves facial details like wrinkles while maintaining the distinct characteristics of the chosen face and hair styles, demonstrating its ability to decompose and recombine these elements effectively.

References

[1] Junxuan Li, Chen Cao, Gabriel Schwartz, Rawal Khirodkar, Christian Richardt, Tomas Simon, Yaser Sheikh, and Shunsuke Saito. Uravatar: Universal relightable gaussian codec avatars. *arXiv preprint arXiv:2410.24223*, 2024. 4

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 1