

Supplementary Materials: E^3 Gen: Efficient, Expressive and Editable Avatar Generation

Anonymous Authors

This is the supplementary material for E^3 Gen:Efficient, Expressive and Editable Avatar Generation. We include a video(Sec 1) summarizing our method and exhibiting more generation and animation results. Implementation details of our method are introduced in Sec 2. We also provide additional qualitative experimental results and limitation discussions in Sec 3.

1 SUPPLEMENTARY VIDEO

We provide a supplementary video for comprehensive analysis of our method. The video includes:

- An overview of our method E^3 Gen;
- Results of unconditional generation and novel view synthesis;
- Results of novel pose animation;
- Results of editing.

2 IMPLEMENTATION DETAILS

2.1 Network Architecture

Geometry and Appearance Decoder. Geometry decoder D_g consists of one shared layer and two separate heads to predict opacity value α and offset value δ_μ for the initial position. Appearance decoder contains one shared layer and one head to predict color c , relative rotation δ_r , and relative scale δ_s . For shared layers, we utilize SiLU as an activation function. And for prediction heads except for offset δ_μ prediction head, Sigmoid function is used. For offset prediction head, we do not use any activation function, instead, we initialize the weights in the head to be $w \sim \mathcal{U}(-1 \times 10^{-5}, +1 \times 10^{-5})$ and the initial biases are set to be 0.

Denoising U-Net. The denoising U-Net is constructed as 2D-UNet with intermediate attention layers, following [1]. Model configuration is provided in Tab 1.

2.2 Data Preparation

We select 500 scans from THuman2.0 [3] Dataset and render 54 camera views for each scan. The horizontal angle is distributed uniformly around the subjects. The rendering code is based on the rendering code of ICON [2]. The focal length is set to be 5000.

2.3 Training Configuration

We represent each generated avatar using 50,200 Gaussian primitives. For each training step, we randomly select 4 camera views for each scene. The total loss for each training step is:

$$\mathcal{L} = \lambda_{\text{fit}} \mathcal{L}_{\text{fit}}(x_i, \psi) + \lambda_{\text{denois}} \mathcal{L}_{\text{denois}}(x_i, \phi), \quad (1)$$

where λ_{fit} , $\mathcal{L}_{\text{denois}}$ are equal to $c_{\text{fit}} \left(1 - e^{-0.1N_v}\right) / N_v$ and $c_{\text{denois}} / \text{EMA} \left(\|x_i\|_F^2\right)$. c_{fit} is set to be 20, while c_{denois} is 20. EMA means exponential moving average. $\|x_i\|_F^2$ is the Frobenius norms

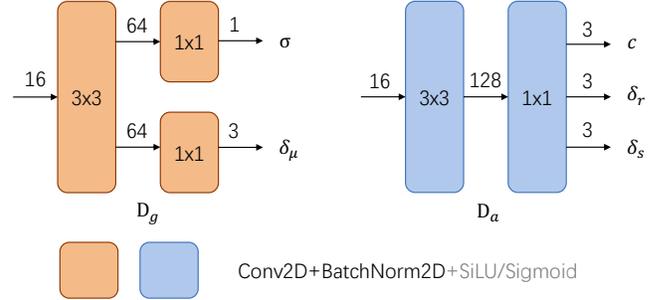


Figure 1: Network Architecture of Decoders. We represent geometry attribute decoder D_g as blocks in orange color and appearance attribute decoder as blocks in blue color. The numbers on the arrow line are the input and output channels for each block. The number on the block denotes the kernel size of convolutional layers in that block. Each block contains one 2D convolutional layer, followed by a batch normalization layer and finally an activation layer. Different blocks adopt different activation functions, which will be introduced in Sec 2.1.

Table 1: Model Configuration.

Key	Value
Number of timesteps	1000
Noise configuration	Linear
Input size	$256 \times 256 \times 32$
Base channel	128
Channels configuration	[0.5, 1, 2, 2, 4, 4]
Resblocks per downsample	2
dropout	0
Number of attention heads	4
Attention resolution	[32, 16, 8]
Downsample method	Conv
Upsample method	Conv

of generative UV features plane x . For fitting process loss:

$$\mathcal{L}_{\text{fit}}(x_i, \psi) = \lambda_c \mathcal{L}_c + \lambda_{\text{vgg}} \mathcal{L}_{\text{vgg}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}. \quad (2)$$

$\lambda_c = 20$, $\lambda_{\text{vgg}} = 0.005$, and $\lambda_{\text{reg}} = 50$.

Our model is implemented using PyTorch, and we utilize the Adam optimizer during the training process. For denoising U-Net, the learning rate is 1×10^{-4} , while for decoders, the learning rate is 1×10^{-3} . The learning rate for generative UV features is 0.04. We set the batch size to be 8 scenes. The model is trained for 150000 iterations, which takes approximately 6 days to complete, utilizing 2 NVIDIA 3090 GPUs.

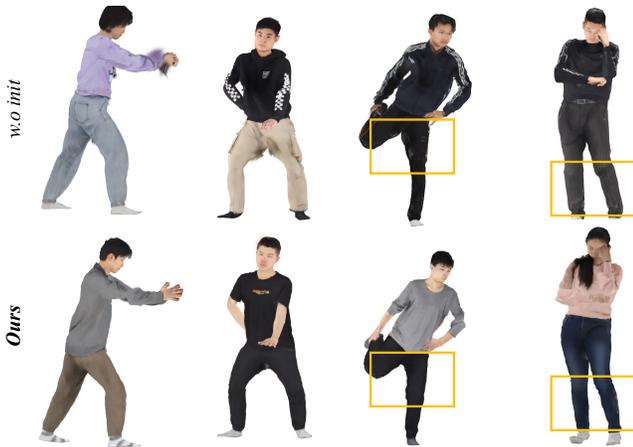


Figure 2: We qualitatively compare randomly generated samples in the same poses. With initialization, our method exhibits fewer spiking artifacts or voids (zooming in for clear observation).

3 ADDITIONAL EXPERIMENTAL RESULTS AND LIMITATIONS

3.1 Qualitative Evaluation for Initialization

We present the random generation results both with and without the initialization process, as depicted in Fig 2. The initialization process

contributes to the generated avatars by providing a smoother surface and enabling stretching capabilities, which effectively reduces the occurrence of voids and spiking artifacts during the animation process.

3.2 Limitations

(1) Although our method can represent loose cloth, animating them remains challenging and can have artifacts. (2) As the generative UV features plane representation initialized with SMPL-X parametric model, inaccurate estimation of parameters of SMPL-X would lead to artifacts and affect the training results. Adding SMPL-X optimization process during training might help to alleviate the impact of inaccurate estimation results. (3) Due to the limitation in the number of samples, overfitting is hard to forbid. Extending our method to a large-scale multi-view video dataset is promising. As these large-scale datasets have not been fully announced, we still train our model on 3D scan datasets.

REFERENCES

- [1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
- [2] Yuliang Xiu, Jinlong Yang, Dimitrios Tzionas, and Michael J. Black. 2022. ICON: Implicit Clothed humans Obtained from Normals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13296–13306.
- [3] Tao Yu, Zerong Zheng, Kaiwen Guo, Pengpeng Liu, Qionghai Dai, and Yebin Liu. 2021. Function4D: Real-time Human Volumetric Capture from Very Sparse Consumer RGBD Sensors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2021)*.