CAP4D: Creating Animatable 4D Portrait Avatars with Morphable Multi-View Diffusion Models

Supplementary Material

This document includes supplementary implementation details and results. We provide implementation details related to the morphable multi-view diffusion model (MMDM), 3D morphable model (3DMM), 4D avatar, datasets, and evaluation procedures; we also provide additional evaluations and ablation studies. We encourage the reader to inspect our visual results, comparisons with other models, and additional visualizations in the accompanying project page: felixtaubner.github.io/cap4d.

A. MMDM Implementation

A.1. Model architecture

Our model is based on Stable Diffusion 2.1 [10] and illustrated in Fig. S1. We remove all cross-attention layers and replace the 2D self-attention layers after 2D residual blocks with 3D attention layers to create the multi-view diffusion model. Following Gao et al. [31], we only modify the 2D self-attention for layers with dimensions 32×32, 16×16, and 8×8. We also adjust the first convolutional layer of the model to accommodate the additional conditioning channels and, where possible, initialize all layers with pre-trained weights.

During training, we update all model parameters and follow Stable Diffusion with the following adjustments. First, we shift the signal-to-noise ratio of the noise schedule by $\log(\sqrt{N})$. Adjusting the noise schedule provides more diffusion steps for the model to learn coarse structures in the generated images [31, 42]. Second, we find that adjusting the noise schedule to have zero terminal SNR is vital to avoid artifacts in the background [56]. Our latent diffusion model has a total of $815M$ parameters. We use a classifier-free guidance weight of 2 during sampling.

A.2. Conditioning

The MMDM takes as input a set of reference or generated images, and each set is paired with five additional sets of conditioning images (illustrated in Fig. 3): $\mathbf{V}_{\text{ref/gen}}$, view direction maps containing per-pixel view directions in world coordinates; $\mathbf{P}_{\text{ref/gen}}$, 3D pose maps with rasterized vertex positions of the 3DMM template mesh; $\mathbf{E}_{\text{ref/gen}}$, expression deformation maps with rasterized vertex deformation vectors; and $\mathbf{B}_{\text{ref/gen}}$, pairs of binary masks that indicate (1) outcropped areas that are padded with white pixels and (2) whether the input is a reference or generated image.

Preprocessing steps and binary masks. To create the reference conditioning images, we first obtain FLAME [53] parameters, camera intrinsic parameters, and extrinsic parameters using the 3DMM estimator of Taubner et al. [85]. To create the generated conditioning images, we sample the FLAME parameters as described in the following section.

We crop the reference images by fitting a bounding box around the vertices of the 3DMM projected onto the camera image plane. Then, we find the smallest square bounding box that encloses the original bounding box (centered at the same location) and enlarge the result by 30% to include the hair, neck, and shoulders; this bounding box is used for cropping. We resize the cropped image to 512×512 resolution, adjust the camera intrinsics to be consistent with this cropped frame, and remove the background using an off-the-shelf background matting model [55].

Sometimes, the bounding box used to crop the image extends outside the image boundaries. To perform outcropping in such regions, we pad the image with white pixels, and we flag these regions using a binary outcropping mask, where all outcropped areas are indicated with white pixels. The MMDM is conditioned on $\mathbf{B}_{\text{ref/gen}}$, consisting of the outcropping masks and binary masks that indicate whether the input is a reference or generated image.

View direction conditioning. We use the camera intrinsic and extrinsic parameters to compute view conditioning images, $\mathbf{V}_{\text{ref/gen}}$, containing the view direction for each pixel in world coordinates. The world coordinates are computed relative to the first reference view, which is positioned at the origin of the coordinate system with its rotation matrix set to the identity.

3D pose and expression deformation conditioning. As described in the main text, we obtain the 3D pose map $\mathbf{P}_{\text{ref/gen}}$ by texturing the vertices of the tracked 3DMM model with the 3D vertex positions of the 3DMM template mesh $\mathbf{T}$. We rasterize these vertex positions and encode the values using a periodic positional encoding [84]:

$$\gamma(p) = (\sin(2^0 p), \cos(2^0 p), \dots, \sin(2^{L-1} p), \cos(2^{L-1} p)), \tag{S1}$$

where p is the 3D vertex position texture, and $L = 7$ is the number of encoding frequencies. This results in 42 positional encoding channels. We compute $\mathbf{E}_{\text{ref/gen}}$ in a similar fashion by rasterizing the 3D deformations caused by the expression blendshape parameters ($\mathcal{E}(\phi)$), but we omit the positional encoding.

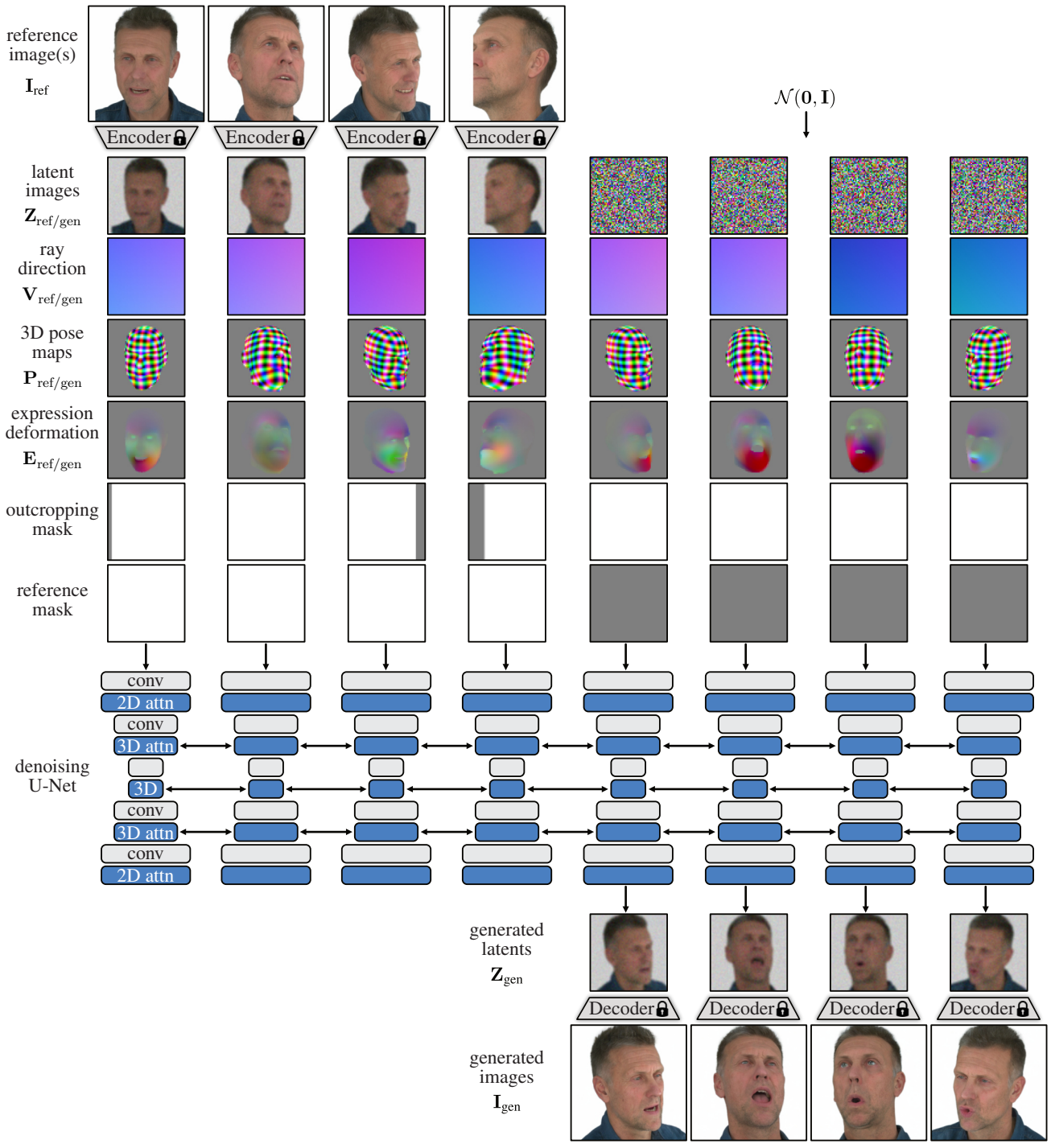

Figure S1. **MMDM architecture.** Our model is initialized from Stable Diffusion 2.1 [10], and we adapt the architecture for multi-view generation following CAT3D [31]. We use a pre-trained image encoder to map the input images into the latent space, and we use the latent diffusion model to process eight images in parallel. We replace 2D attention layers after 2D residual blocks with 3D attention to share information between frames. The model is conditioned using images that provide information such as head pose, expression, and camera view. The denoised latent image is decoded using a pre-trained decoder.

A.3. MMDM sampling

We follow a fixed sampling procedure to obtain novel generated views and 3DMM parameters as illustrated in Fig. S2. We begin by sampling a set of G generated camera views, where each view is rotated around the center of the head with a randomly sampled azimuth ψ and elevation angle θ (we set the view aligned straight on with the face to have zero azimuth and zero elevation). The camera is kept at the same distance from the head as the first reference view. The values ψ and θ are uniformly sampled to be within an ellipse (red line, Fig. S2):

$$(\frac{\psi}{\psi_{\text{max}}})^2 + (\frac{\theta}{\theta_{\text{max}}})^2 < 1, \tag{S2}$$

where $\psi_{\text{max}} = 55°$ and $\theta_{\text{max}} = 20°$.

Expression database. We select a unique expression parameter for each camera view from our expression database. The database is created using a diversity-promoting sampling scheme (implemented in the `diversipy` software package [77]) that partitions the space of expressions obtained from all frames of the Nersemble [50] dataset into $G = 840$ dissimilar subsets with a representative sample for each subset. To determine the distance between each expression sample, we use Euclidean distance in expression parameter space $\boldsymbol{\phi} \in \mathbb{R}^{65}$, where each dimension is weighted by the maximum vertex displacement of the corresponding blendshape.

B. FLAME 3DMM Implementation

The FLAME representation [53] consists of $N_v = 5023$ vertices, which are controlled by identity shape parameters $\boldsymbol{\beta}$, expression shape parameters $\boldsymbol{\phi}$, and skeletal joint poses through linear blend skinning. We ignore the jaw pose and use the FLAME2023 model, which includes deformations due to jaw rotation within the expression blendshapes. Overall, there are $\boldsymbol{\beta} \in \mathbb{R}^{150}$ identity shape parameters and $\boldsymbol{\phi} \in \mathbb{R}^{65}$ expression parameters. To model eye rotation, we use one joint rotation for both eyes. Each vertex position is determined by adding expression and identity shape offsets to the template mesh $\mathbf{T}$, and the offsets are computed using the expression and identity shape parameters and the corresponding linear bases, $\mathcal{E}$ and $\mathcal{S}$:

$$\mathbf{m} = \mathbf{T} + \mathcal{E}(\boldsymbol{\phi}) + \mathcal{S}(\boldsymbol{\beta}). \tag{S3}$$

We use an edited version of the FLAME template mesh to create the conditioning signals used by the MMDM. More precisely, we manually position a spherical mesh inside the mouth region and behind the lip to represent the upper jaw. This sphere is static and unaffected by the expression shape parameters $\boldsymbol{\phi}$. We remove the lower neck vertices and limit the conditioning model to the head region.

For the representation used by the 4D avatar, we add a spherical mesh to model the lower jaw. This sphere is placed similarly to the upper jaw mesh, but it is rigged to move with the jaw joint. We compute the jaw rotation heuristically by tracking the deformation of a specific vertex on the lower jaw relative to the jaw joint position obtained from the FLAME model. We adopt the UV mapping provided by previous work [28] and modify it manually to add textures for the upper and lower jaw meshes.

C. 4D Avatar Implementation

Our 4D avatar model is based on GaussianAvatars [71] with a few modifications to make it more robust to generated views. We describe these changes in the following and illustrate the approach in Fig. S3.

Deformation model. We disable the per-frame finetuning of FLAME parameters used by GaussianAvatars during training as we find that it leads to overfitting. To correct inaccuracies in the underlying 3DMM, we use a U-Net to deform the mesh with expression-dependent deformations.

The input to the U-Net consists of UV maps that encode the expression deformations and positional encodings of UV map pixel locations. To compute the expression deformation map, we first remesh the FLAME head to achieve pixel-aligned vertices in UV space at a 128×128 resolution. Then, we rasterize the deformations caused by the expression parameters $\mathcal{E}(\phi)$ into UV space. We obtain a positional encoding of the UV space by encoding the UV coordinates with the same periodic functions as in Eq. (S2), where we set the number of frequencies to $L = 6$ and the coordinate p to the UV-space coordinate of each pixel, leading to a total number of 24 encoding channels. The positional encoding is concatenated to the UV-space expression deformation and processed by a 6-layer U-Net [75].

The U-Net outputs a 3-channel deformation map, $\mathbf{D}_{\text{uv}}$, (see Fig. S3) indicating the expression-dependent deformation correction. We mask this deformation to prevent deformations in static areas such as the back of the head and lower neck. To obtain the final vertex positions, we add these deformations to the vertices produced by the FLAME model.

During training, we use multiple regularizers to prevent motion artifacts. First, we apply a weight decay of 2×10^{-3} on the U-Net weights; second, we use an L2 loss $\mathcal{L}_{\text{lap}} = ||\Delta \mathbf{D}_{\text{uv}}||_2^2$ on the Laplacian of the deformation map; last we append an L2 loss on the relative deformation and rotation of each Gaussian $\mathcal{L}_{\text{deform}}$ and $\mathcal{L}_{\text{rot}}$. We logarithmically decrease the learning rate of this network from 10^{-5} to 10^{-7} during training.

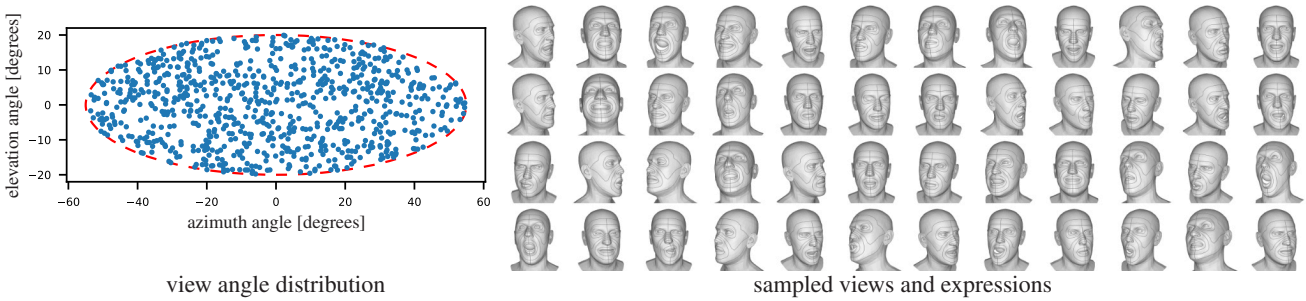

Figure S2. **3DMM Sampling.** To generate novel views, we uniformly sample in azimuth and elevation (left). For each camera view, we select unique expression parameters from our expression database, which is obtained from the Nersemble dataset [50] following a diversity-promoting sampling scheme [77]. A subset of the sampled expressions and views is visualized on the right.

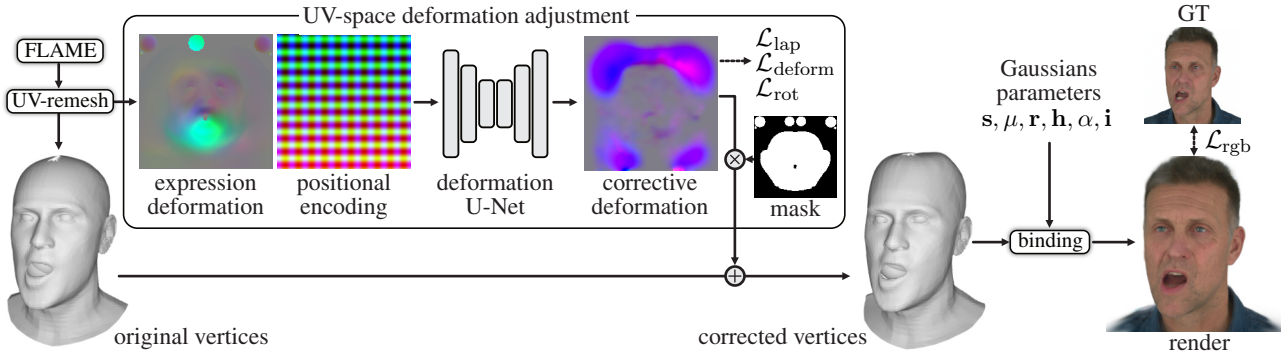

Figure S3. **Overview of 4D Avatar Model.** Our 4D representation incorporates multiple improvements to the GaussianAvatars [71] model. First, we re-mesh the FLAME topology so that each vertex corresponds to a pixel in the UV space. Then, we input the UV-space deformations caused by the expression blendshapes and a UV-space positional encoding into a deformation U-Net. This U-Net outputs corrective deformations, which, after masking, are added to the remeshed FLAME output. Following Qian et al. [71], the Gaussians are parameterized by a scale $\mathbf{s}$, local position μ, local rotation $\mathbf{r}$, spherical harmonics coefficients $\mathbf{h}$, opacity α, and parent triangle $\mathbf{i}$. We apply regularizers to the output of the U-Net, and we add an LPIPS penalty to the photometric loss $\mathcal{L}_{\text{rgb}}$.

LPIPS loss. To make the reconstruction more robust to inconsistencies in the generated views, we add an LPIPS [110] loss to the existing photometric loss from GaussianAvatars [71] and weight it against the other term:

$$\mathcal{L}_{\text{rgb}} = \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}} + (1 - \lambda_{\text{LPIPS}}) \mathcal{L}_{\text{rgb,GA}}, \tag{S4}$$

where λ_{LPIPS} is the weighting of LPIPS loss, which we linearly increase from 0 to 0.9 during training. $\mathcal{L}_{\text{rgb,GA}}$ is the original photometric loss from GaussianAvatars. We also include their scaling and positional losses $\mathcal{L}_{\text{scaling}}$ and $\mathcal{L}_{\text{position}}$, resulting in the modified total loss function:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{deform}} \mathcal{L}_{\text{deform}} + \lambda_{\text{rot}} \mathcal{L}_{\text{rot}} + \mathcal{L}_{\text{scaling}} + \mathcal{L}_{\text{position}}, \tag{S5}$$

where $\lambda_{\text{deform}} = 0.4$ and $\lambda_{\text{rot}} = 0.005$ are the weights for the corresponding losses. For more information on these loss functions, we refer to GaussianAvatars [71].

Other changes. We attach the Gaussians to the triangles of the re-meshed FLAME model. Each Gaussian contains a scale $\mathbf{s}$, local position μ, local rotation $\mathbf{r}$, spherical harmonics coefficients $\mathbf{h}$, opacity α and parent triangle $\mathbf{i}$. We initialize the avatar with $100k$ Gaussians, where the number of Gaussians for each triangle is proportional to the area of the triangle. Also, we set each Gaussian's initial scale to be inversely proportional to the number of Gaussians per triangle, which we find to reduce rendering artifacts.

D. Datasets

We train the MMDM using the monocular video dataset VFHQ [94], and the multi-view datasets Nersemble [50], MEAD [90] and Ava-256 [64]. For MEAD, we use the sequences with neutral emotions. For Ava-256, we randomly select 20 sequences with 16 forward-facing camera views for each subject. We jointly estimate 3DMM parameters, camera extrinsics, and intrinsics using a multi-view face tracker [85]. For Nersemble and Ava-256, we use the available camera calibration. We remove frames where less than 95% of the head is visible. For VFHQ, we detect and

	single reference image			
Method	SSIM ↑	AED ↓	APD ↓	AKD ↓
Voodoo3D [88]	0.658	1.463	0.085	10.52
GAGAvatar [19]	0.718	1.076	0.069	13.01
Real3D [102]	0.667	1.561	0.118	15.91
Portrait4D-v2 [23]	0.651	1.291	0.121	19.97
MMDM only	0.730	**0.707**	**0.041**	5.82
CAP4D	**0.748**	0.782	**0.041**	**5.68**

	10 reference images			
Method	SSIM ↑	AED ↓	APD ↓	AKD ↓
DiffusionRig [24]	0.624	0.930	0.084	17.8
FlashAvatar [93]	0.580	1.243	0.124	20.8
GaussianAvatars [71]	0.628	1.413	0.237	21.5
no MMDM	0.590	1.064	0.093	10.6
MMDM only	**0.753**	**0.542**	**0.033**	**5.09**
CAP4D	0.748	0.782	0.041	5.68

	100 reference images			
Method	SSIM ↑	AED ↓	APD ↓	AKD ↓
DiffusionRig [24]	0.617	0.975	0.085	18.6
FlashAvatar [93]	0.741	0.689	0.042	6.16
GaussianAvatars [71]	0.713	0.737	0.062	9.01
no MMDM	0.675	0.653	0.058	7.88
MMDM only	0.754	**0.535**	**0.032**	**5.07**
CAP4D	**0.763**	0.634	0.035	5.21

Table S1. **Additional single-image (top) and multi-image (middle, bottom) self-reenactment metrics.** Our method consistently outperforms baselines in terms of expression accuracy (AED), photometric quality (SSIM), and alignment accuracy (AKD).

Method	AED ↓	APD ↓
Voodoo3D [88]	2.428	0.082
GAGAvatar [19]	**2.137**	0.079
Real3D [102]	2.581	0.104
Portrait4D-v2 [23]	2.084	**0.071**
Ours	2.138	0.089

Table S2. **Additional cross-reenactment metrics.** We report additional metrics for our cross-reenactment evaluation.

remove videos with scene changes by checking the acceleration of the keypoints detected using the face tracker. Also, we use MediaPipe [109] to detect and remove frames containing hands, and we use the face tracker to detect and remove frames containing multiple faces. We estimate the gaze direction using a gaze estimation model [1] and convert it to the eye rotation of the FLAME model. In the multi-view sequences, we use the most forward-facing view to estimate the eye gaze. During training, we randomly select R' reference images and G' target images from all views and frames within a sequence with equal probability.

E. Evaluation

In this section, we provide details on our implementation, additional evaluation metrics, and additional ablations.

E.1. Implementation

We use the same predicted FLAME parameters to evaluate our method and our implementations of FlashAvatar [93] and GaussianAvatars [71]. The FLAME expression and identity shape parameters are extracted for each set of reference images [85]. Then, we fit expression parameters to the target frames while preserving the identity parameters.

All metrics for self-reenactment are measured after center-cropping to the head region, resizing to 512×512 resolution, and removing the background using masks included in the Nersemble dataset.

For cross-reenactment, we orbit the camera around the head to allow a better assessment of 3D geometry. This orbit follows an elliptic trajectory defined by the maximum azimuth and elevation angles of $\pm 55°$ and $\pm 20°$ relative to the center of the head. The camera performs one rotation around the head every 8 seconds.

Figure S4. **Qualitative results on RenderMe-360.** Rendered novel views for a 80° view angle change compared to a single ref. image (left) and 5° outside the convex hull of view angles for 100 ref. images (right). Our method outperforms the baselines (please zoom in).

	single reference image			
Method	PSNR↑	LPIPS↓	CSIM↑	JOD↑
F-Y-Emoji [61]	16.67	0.297	0.431	4.35
GAGAvatar [19]	20.50	0.225	0.428	5.83
MMDM only	**21.73**	**0.191**	**0.617**	6.01
CAP4D	21.01	0.212	0.490	**6.08**

	10 reference images				100 reference images			
Method	PSNR↑	LPIPS↓	CSIM↑	JOD↑	PSNR↑	LPIPS↓	CSIM↑	JOD↑
FlashAvatar [93]	23.12	0.251	0.457	5.96	24.32	0.208	0.628	6.60
GaussianAvatars [71]	20.95	0.282	0.573	5.46	24.24	0.219	0.722	6.61
MMDM only	**25.21**	**0.144**	**0.794**	**6.96**	**25.31**	**0.144**	**0.799**	**6.98**
CAP4D	24.58	0.171	0.722	6.74	24.47	0.171	0.733	6.73

Table S3. **Self-reenactment results on the RenderMe-360 dataset.** CAP4D outperforms previous methods across all metrics.

E.2. Additional Metrics

We provide results with the structural similarity metric SSIM, and following previous work [19, 51], we evaluate the average expression distance (AED) and average pose distance (APD) predicted using DECA [28] for both self and cross-reenactment. For cross-reenactment, only the jaw pose distance is used. We measure the average keypoint distance (AKD) using facial landmarks predicted from a keypoint detector [13]. We report these additional metrics for self-reenactment with different numbers of reference images in Tab. S1, and for cross-reenactment in Tab. S2. The metrics show that our approach outperforms the baselines for the self-reenactment task. For cross-reenactment, our method and the baselines all have similar quantitative scores, but our approach shows clear improvements, as demonstrated in the user study and qualitative results.

E.3. User Study

For each video pair, we ask each participant questions to assess their preference in the following criteria using the following prompts.

- **Visual Quality (VQ):** Evaluate the clarity and visual appeal of each video. Your assessment should focus on the face and head region and ignore the neck and upper body.
- **Expression Transfer (EQ):** Determine which generated avatar's facial expressions better match the driving video.
- **3D Structure (3DS):** Assess how well the 3D structure of the head is preserved across different viewing angles and expressions.
- **Temporal Consistency (TC):** Examine how smoothly and naturally the avatar maintains consistent appearance, expression, and movement across consecutive video frames.
- **Overall Preference (OP):** State your overall preference between the two videos. This is your subjective appraisal of which avatar, in your view, performs better.

The video pairs are presented in a random order, and each participant is asked to select either the left or right video for each criterion. We collect a total number of 4800 responses from 24 participants and conduct χ^2-tests to evaluate statistical significance at the $p < 0.05$ level.

E.4. Self-reenactment on RenderMe-360

We conduct additional self-reenactment experiments using the RenderMe-360 dataset [67]. We follow Sec. 5.1 and randomly sample 9 subjects (one sequence per subject). We select the 19 out of 60 camera views from the dataset within the same view angle range as our Nersembled evaluation set (Sec. 5.1). From these 19 views, we hold out eight random views for evaluation (with 25 frames each) and sample reference images from the remaining viewpoints. For this evaluation, we add the video-based baseline Follow-Your-Emoji [61]. Since this is a video-based method, we generate one video for each view. The quantitative (Tab. S3) and qualitative (Fig. S4) results follow the trends in the paper: CAP4D also outperforms previous methods on this dataset.

E.5. Additional Results

We provide additional self-reenactment and cross-reenactment results in Figs. S5 and S6. For self-reenactment, we directly compare our method to baselines with one, ten, or 100 reference images. As the number of reference images increases, our approach (both MMDM-only and CAP4D) improves in quality. With 100 reference images, we recover fine details in the hair and blemishes on the face. Our approach better preserves the identity and exhibits higher visual fidelity compared to baselines in the case of one reference image. We improve photorealism compared to baselines using ten or 100 reference images. The cross-reenactment results show trends similar to those in the main text. Compared to baselines, CAP4D better preserves the identity and fine details of the hair, face, and attire.

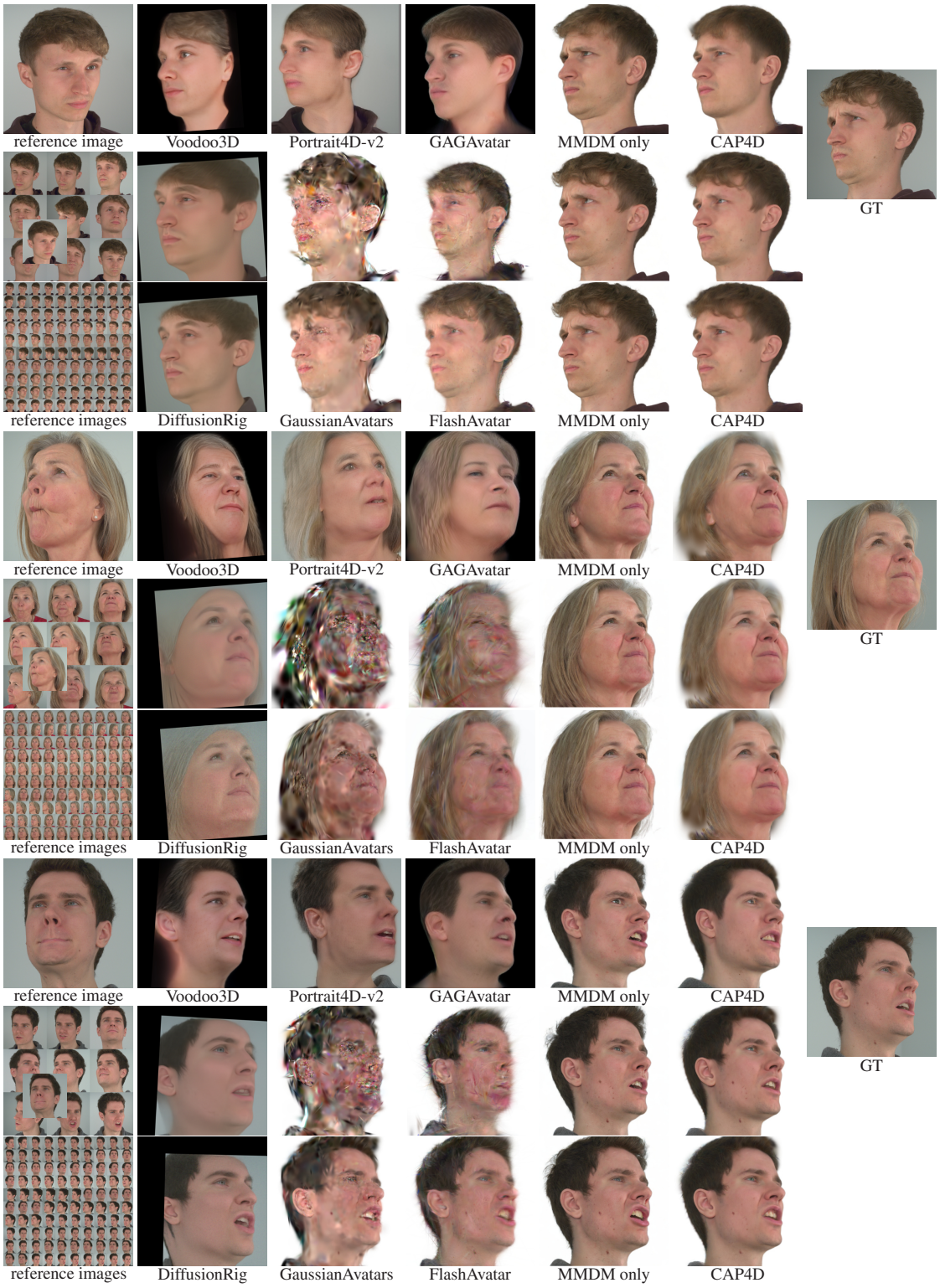

Figure S5. **Self-reenactment results.** We show more qualitative results for our self-reenactment evaluation with varying numbers of reference frames. Both our MMDM and final 4D avatar can leverage additional reference images to produce details that are not visible in the first reference image (hair, top three rows: birthmarks, last three rows). Our results are significantly better compared to previous methods, especially when the view direction differs greatly from the reference image.

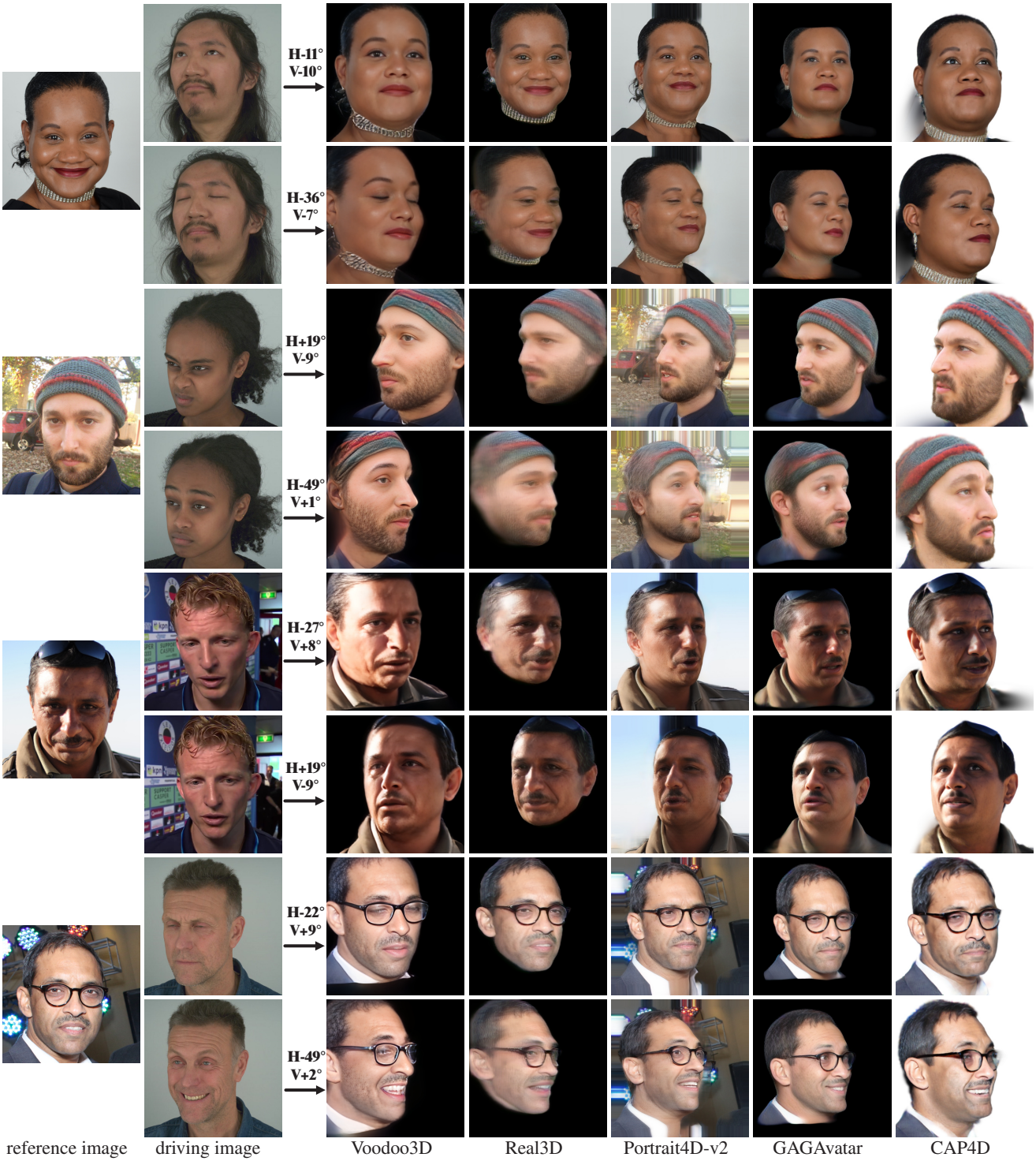

Figure S6. **Cross-reenactment results.** We show additional qualitative results of our cross-reenactment evaluation. We show generated frames under different driving expressions and viewing angles. Our method consistently produces 4D avatars of higher visual quality and 3D consistency even across challenging view deviations. Our avatar can also model realistic view-dependent lighting changes (rows 5 and 6). Best viewed zoomed in.

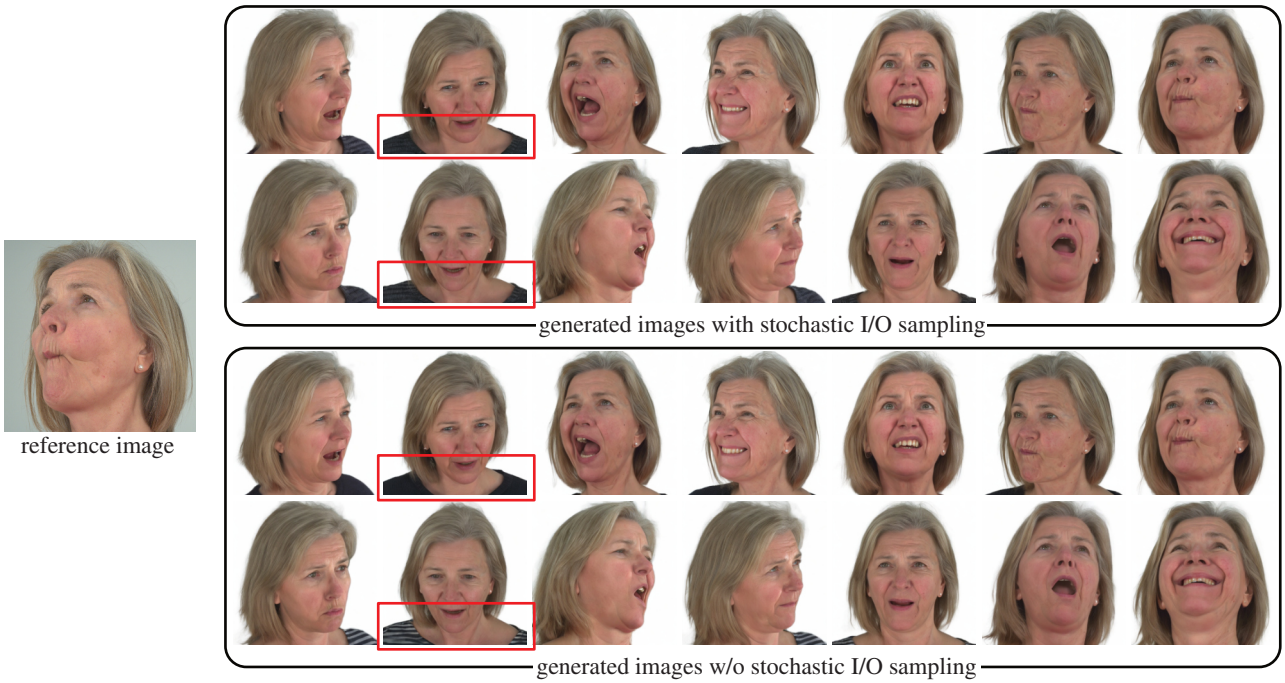

Figure S7. **Ablations on stochastic I/O sampling.** We generate 14 images from a single reference image (left). Parts of the body do not appear in the reference image (shirt) and are thus ambiguous. When we generate these images without stochastic I/O sampling, two batches of seven images are generated separately (i.e., the third and last rows). This results in visible inconsistencies, such as a changing shirt pattern (red boxes). Stochastic sampling (first and second row) generates these images with information shared across all frames, resulting in a consistent shirt appearance.

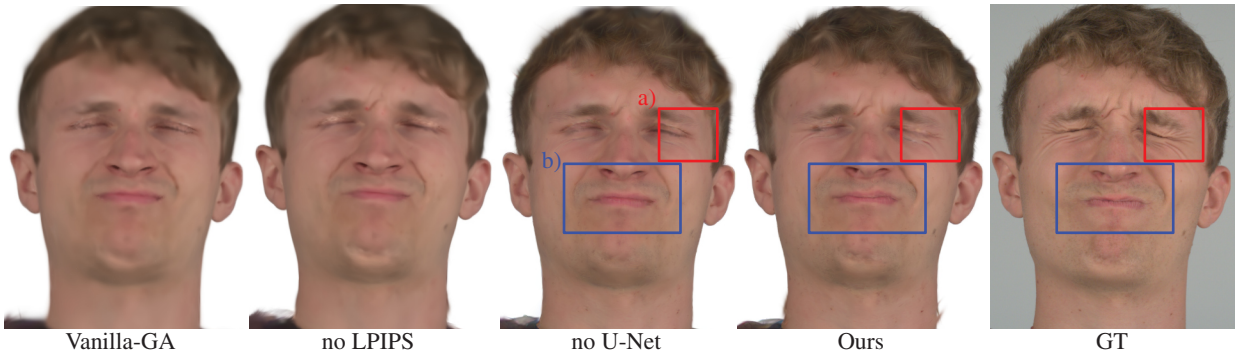

Figure S8. **Ablations on 4D avatar.** We show qualitative results with ablations of our 4D avatar for self-reenactment, using 10 reference images and images generated using the MMDM. From the left: The original implementation of GaussianAvatars without modifications (Vanilla-GA), our model without LPIPS loss (no LPIPS), our model without the U-Net correcting deformations (no U-Net), our final version (Ours), and the ground truth (GT). Without LPIPS, the avatar appears significantly blurrier. With the U-Net deformations, dynamic details such as wrinkles are depicted more accurately (wrinkles in (a)), and expressions are depicted overall more accurately (lips and nasolabial fold in (b)).

Category	Ablation	PSNR ↑	LPIPS ↓	CSIM ↑	JOD ↑
sampling (single ref.)	w/o stochastic	21.60	0.325	0.625	5.31
	Ours	21.82	0.317	0.632	5.40
4D rep.	w/o stochastic	21.50	0.320	0.624	5.60
	$G = 420$	21.37	0.328	0.620	5.48
	Ours ($G = 840$)	21.69	0.311	0.633	5.67
	$G = 1260$	21.71	0.313	0.632	5.69

Table S4. **Additional ablations.** We assess the impact of our stochastic I/O conditioning with a single reference image. Also, we show the impact of stochastic I/O conditioning and the number of generated images on the 4D reconstruction.

Ablation	PSNR ↑	LPIPS ↓	CSIM ↑	JOD ↑	time (min)
R'=3, G'=5	23.72	0.277	0.793	5.99	224
R'=4, G'=4	23.82	0.270	0.804	6.06	298
R'=5, G'=3	23.94	0.264	0.810	6.13	374

Table S5. **Ablation study on the number of ref. images R'.**

E.6. Additional Ablation Study

Stochastic I/O sampling. We conduct experiments with and without the stochastic I/O sampling on the self-reenactment task with a single reference image, and report the results in Tab. S4. With only a single reference image, stochastic sampling improves image quality (PSNR and LPIPS) and consistency between frames (JOD).

In Fig. S7, we generate an example set of 14 images. Inconsistencies such as changing shirt patterns appear without stochastic sampling, whereas using stochastic sampling improves overall consistency.

4D avatar. We conduct additional ablations on the ten-reference-image self-reenactment task (see Fig. S8 and Tab. S4); we show the impact of our improvements to the GaussianAvatars representation [71] and the impact of stochastic I/O sampling and varying the number of generated images on the quality of the 4D avatar. Qualitatively, adding the LPIPS loss and expression-dependent deformations predicted by the U-Net improves the ability to reconstruct wrinkles and expression-dependent details (Fig. S8).

The number of generated images also affects the quality of the avatar. Specifically, we evaluate generating $G = 420$, 840, or 1260 images and reconstructing the avatar. When generating fewer images, we observe worse reconstruction (LPIPS); adding additional images beyond $G = 840$ does not significantly improve the results, but requires additional compute. We also find that stochastic sampling improves the 4D avatar in terms of PSNR, LPIPS, CSIM, and JOD.

Number of reference images in the MMDM. We ablate the number of reference images used in the MMDM for each forward pass (Sec. 3.1, Fig. S1) in Tab. S5. We use $R' = [3, 4, 5]$ (and $G' = [5, 4, 3]$) and show that a higher R' (lower G') leads to better results on self-reenactment with 10 reference images, but requires more compute/forward passes to generate the 400 evaluation frames (Algorithm 1). Results in the paper use $R' = G' = 4$.

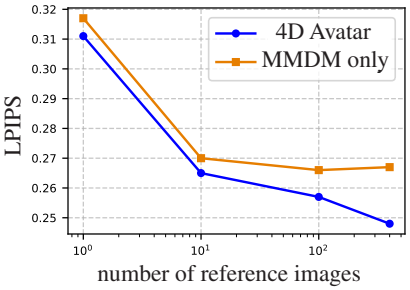

Figure S9. **Analysis of Reference Quantity.** We observe that with hundreds of reference images, the performance of the MMDM plateaus, while our 4D avatar continues scaling with the input quantity. This shows that the 4D avatar can seamlessly benefit from both generated and reference images.

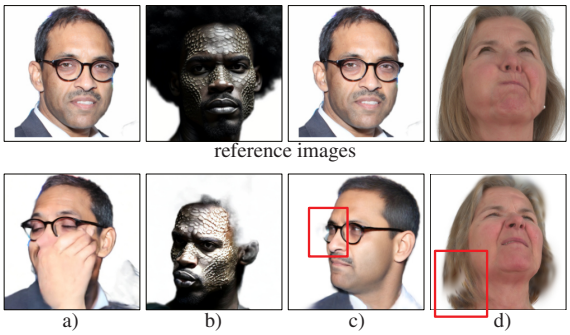

Figure S10. **Failure Cases.** (a) Our training dataset contains some images that were not properly filtered out using our automated pipeline, and so the MMDM sometimes generates images where the face is occluded (e.g., by a hand). (b) Some training images contain a faulty background segmentation, occasionally leading to artifacts in the MMDM output. (c) Our 4D avatar can fail in areas not modeled by the FLAME topology, such as glasses and (d) long hair.

More total reference images. Finally, we conduct an additional experiment with 400 reference images, which we evaluate in the same way as the previous experiments on the self-reenactment task. Evaluations (plotted in Fig. S9) show that while the performance of our MMDM plateaus, the final 4D avatar can leverage and scale with hundreds of reference images. This is likely because the 4D avatar can improve its reconstruction from the reference images through the direct optimization procedure.

E.7. Failure Cases

We show failure cases for the MMDM and the 4D avatar in Fig. S10. Specifically, we find that some generated images reflect imperfections in our training dataset, such as images where a hand occludes the face, or images where there are artifacts due to imperfect background segmentation. Also, the 4D avatar cannot model certain regions perfectly, such as hair or glasses, since these are not modeled in the FLAME topology.