We will release the metadata, crops, camera parameters, landmarks, and raw images from our dataset, as well as the data processing code, training code, FID evaluation code, and all trained models.

1. Supplementary

In this supplement, we first introduce *Unsplash-Pexels*, an extra large-pose face dataset to facilitate more future work (Sec. 1.1). We then discuss the limitations of our models (Sec. 1.2). We demonstrate that our models can eliminate the "seam" artifact in EG3D in Sec. 1.3. We discuss the problem of the original FID of EG3D (Sec. 1.4). Then we illustrate the influence of the buggy tri-plane in Sec. 1.5. After that, we show our data processing pipeline (Sec. 1.6) and the details of pose density computation (Sec. 1.7), the divided EG3D model (Sec. 1.8), and examples from our *LPFF* dataset (Sec. 1.9). Finally, we provide further visual results of StyleGAN2-ada (Sec. 1.10) and EG3D models (Sec. 1.11) trained on our datasets.

1.1. Unsplash-Pexels Dataset

Besides the images from the Flickr[1], we additionally collect **19,321** images from Unsplash[2] and Pexels[3], and process them using the same image processing method as *LPFF*. We denote the obtained dataset as *Unsplash-Pexels* (see samples in Fig. 1). We combine *Unsplash-Pexels* with *FFHQ*, named *FFHQ+Unsplash-Pexels*.

Figure 1: Representative samples from *Unsplash-Pexels*.

We use the same training methods as E^{Ours}_{var1}, E^{Ours}_{var2} to train models on the *FFHQ+Unsplash-Pexels* dataset, and get models E^{UspPex}_{var1}, E^{UspPex}_{var2}.

However, most of the images in Unsplash and Pexels were taken by professional photographers and processed by image filters, while the images in Flickr were from everyday life scenes taken by Flickr users. As a result, although the images in *Unsplash-Pexels* are processed using the same method as *LPFF* and their camera distributions are similar (Fig. 2), the domain gap between *Unsplash-Pexels* and *FFHQ* would prevent the models trained on *FFHQ+Unsplash-Pexels* from generating view-consistent results.

As shown in Tab. 1 and 2, the models trained on *FFHQ+Unsplash-Pexels* present much worse performance in facial identity consistency and geometry consistency than the models trained on *FFHQ+LPFF*.

Thus we do not use *Unsplash-Pexels* as our training data, but propose this dataset to inspire and facilitate more work in the future.

[1]https://www.flickr.com
[2]https://unsplash.com
[3]https://www.pexels.com

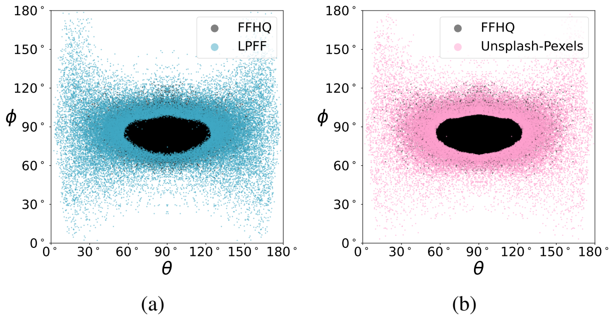

Figure 2: (a) *FFHQ+LPFF*. (b) *FFHQ+Unsplash-Pexels*.

model	$c_g = c_{avg}$ $c_r \sim$ FFHQ	$c_g \sim$ FFHQ $c_r \sim$FFHQ	$c_g \sim$ LPFF $c_r \sim$FFHQ
E^{UspPex}_{var1}	0.800	0.788	0.774
E^{Ours}_{var1}	**0.804**	**0.792**	**0.778**
E^{UspPex}_{var2}	0.774	0.773	0.764
E^{Ours}_{var2}	**0.789**	**0.784**	**0.771**

Table 1: Quantitative evaluation of facial identity consistency (↑).

model	$c_g = c_{avg}$ $c_r \sim$ FFHQ	$c_g \sim$ FFHQ $c_r \sim$FFHQ	$c_g \sim$ LPFF $c_r \sim$FFHQ
E^{UspPex}_{var1}	0.135	0.135	0.142
E^{Ours}_{var1}	**0.119**	**0.124**	**0.134**
E^{UspPex}_{var2}	0.137	0.128	0.134
E^{Ours}_{var2}	**0.117**	**0.122**	**0.131**

Table 2: Quantitative evaluation of geometry consistency (↓).

1.2. Limitations

Our models frequently generate "mask" artifacts at faces (Fig. 3 (a)), resulting in unsmooth cheek surfaces. We discover that this is due to improperly estimated densities. We expect that future research incorporating genuine 3D face priors into the model will aid in resolving this issue. Our models tend to generate holes in faces when rendering eyeglasses because of lens refraction (Fig. 3 (b)). There are also fuzzy results or holes when rendering eyes from extreme camera postures (Fig. 3 (c)). Although our method can generate almost full-head 3D representations, the lack of data on the back of the head leads to incomplete head geometry (Fig. 3 (d)).

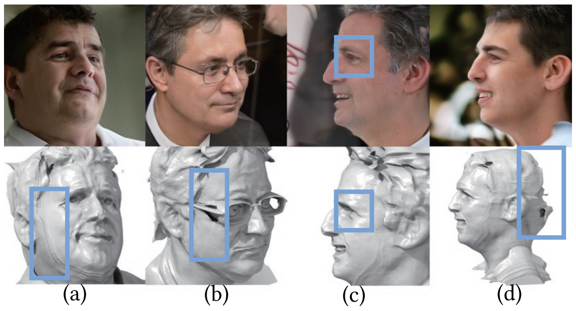

Figure 3: The limitations in our EG3D models (highlighted by blue boxes). (a) "Mask" artifacts. (b) Holes in faces when rendering eyeglasses. (c) Blurry results and holes in eyes. (d) Incomplete head geometry.

Inspired by EG3D, we use [3] to measure the probability of smiling against θ coordinate in *FFHQ+LPFF*. Fig. 4 shows that people typically smile when they are facing the camera, so the models trained on our dataset have the smile-posture entanglement.

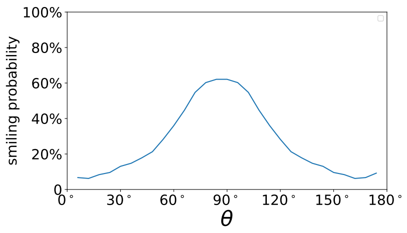

Figure 4: The probability of smiling against θ coordinate in *FFHQ+LPFF*. $\theta = 90°$ stands for a frontal face.

1.3. Seam Artifacts Elimination

Fig. 5 presents the illustration of "seam" artifacts in the results of E^{FFHQ}_{var1}. Our model E^{Ours}_{var1} is trained without requiring any additional regularization loss or any data rebalance strategy, and is free from the "seam" artifacts.

1.4. FID

During our evaluation of the FID for the EG3D model, we found that EG3D computes the FID by conditioning the model on c_g and then rendering results from $c_r = c_g$. In this scenario, as shown in Fig. 6, the generator always perceives the true pose of the rendering camera, resulting in high-quality synthesized images. However, if we set $c_r \neq c_g$, the output exhibits artifacts and distortions. Consequently,

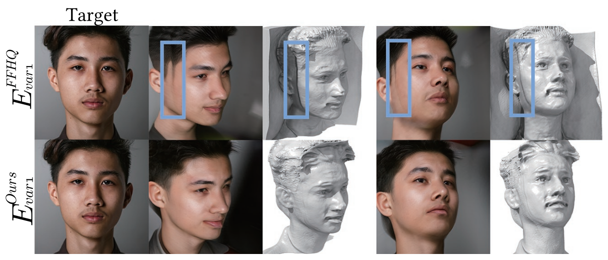

Figure 5: After employing PTI[2] GAN inversion, we use E^{FFHQ}_{var1} (Top) and E^{Ours}_{var1} (Bottom) to render novel views for the same target image. The "seam" artifacts are highlighted by blue boxes in the results of E^{FFHQ}_{var1}.

the original FID measurement fails to accurately assess the quality of overall head geometry and multi-view rendering quality. Thus, we propose sampling c_r and c_g from different distributions.

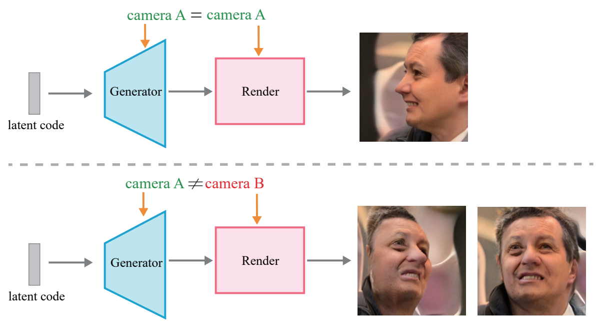

Figure 6: Visualization of the problem in the original FIDs measurement. When $c_r = c_g$ (Top), EG3D outputs good-quality results. When $c_r \neq c_g$ (Bottom), the results exhibit artifacts and distortion.

1.5. Buggy Tri-plane

In our evaluation results for the EG3D model, we found that there was an increased FID when computing $c_g = c_{avg}/c_r, c_r \sim FFHQ$, and we attributed this to the changed data variance and the buggy (XY, XZ, ZX) plane used in E^{FFHQ}_{var1} and E^{FFHQ}_{var2}. To provide a fairer comparison, we retrain the EG3D model using the (XY, XZ, ZY) plane (which was used in our models) on the *FFHQ* dataset. Here, all the training parameters are identical to those of E^{FFHQ}_{var1} and E^{Ours}_{var1}. We denote the obtained model as $E^{FFHQ}_{var1-fixed}$. Then we compare the performance of E^{FFHQ}_{var1}, $E^{FFHQ}_{var1-fixed}$, and E^{Ours}_{var1}. Tab. 3, 4, and 5 show the comparison results, Figs. 7 presents the samples generated by $E^{FFHQ}_{var1-fixed}$.

The only difference between $E^{FFHQ}_{var1-fixed}$ and E^{Ours}_{var1} is their training datasets, but E^{Ours}_{var1} exhibits improvements in FID in most cases. When computing the FID of $c_r \sim$

$FFHQ$, we obtain comparable results for $E^{FFHQ}_{var1-fixed}$ and E^{Ours}_{var1}. In terms of view-consistency, E^{Ours}_{var1} presents comparable performance to $E^{FFHQ}_{var1-fixed}$ in facial identity consistency, and shows improvements in geometry consistency across different sample strategies and datasets.

model	$c_g = c_{avg}$ $c_r \sim$ FFHQ	$c_g \sim$ FFHQ $c_r \sim$FFHQ	$c_g \sim$ LPFF $c_r \sim$FFHQ
E^{FFHQ}_{var1}	0.771	0.768	0.760
$E^{FFHQ}_{var1-fixed}$	0.799	**0.794**	**0.779**
E^{Ours}_{var1}	**0.804**	0.792	0.778

Table 3: Quantitative evaluation of facial identity consistency (↑).

model	$c_g = c_{avg}$ $c_r \sim$ FFHQ	$c_g \sim$ FFHQ $c_r \sim$FFHQ	$c_g \sim$ LPFF $c_r \sim$FFHQ
E^{FFHQ}_{var1}	0.134	0.133	0.159
$E^{FFHQ}_{var1-fixed}$	0.125	0.125	0.144
E^{Ours}_{var1}	**0.119**	**0.124**	**0.134**

Table 4: Quantitative evaluation of geometry consistency (↓).

In sum, our *LPFF* dataset can help the EG3D model to achieve higher image quality on large pose data while not harming the performance on *FFHQ*, and achieve better geometry consistency.

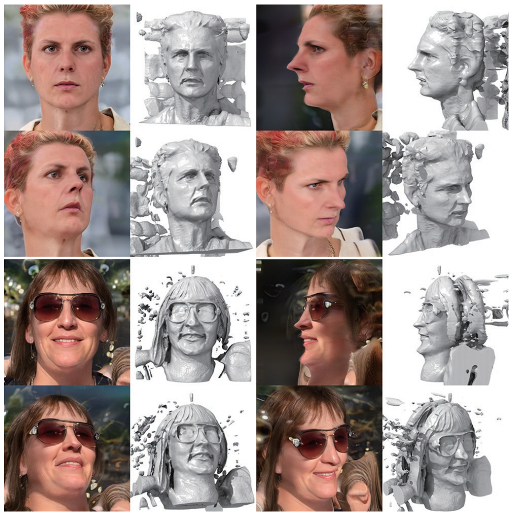

Figure 7: Image-shape pairs produced by $E^{FFHQ}_{var1-fixed}$. We apply truncation with $\psi = 0.8$.

1.6. Data Processing Pipeline

We present the comparison between the image processing pipelines of StyleGAN, EG3D, and ours in Fig. 10.

1.7. Pose Density Computation

We propose representing the 6DOF camera pose and computing the pose density using 2DOF θ and ϕ angles. Because all cameras in EG3D are assumed to be on a spherical surface with radius $r = 2.7$, camera positions are determined by θ and ϕ angles. To prevent faces from appearing outside the image region, the faces are placed near the original point, just like the forward vector points (or look-at points). As a result, the camera's forward direction is determined primarily by its position. We also ignore the upward direction of the cameras because all of the images are rotated and aligned in the preprocessing step using landmarks.

We define Cartesian coordinates as $[r\cos(\theta), r\sin(\theta)\cos(\phi), r\sin(\theta)\sin(\phi)]$ in this paper. Then we compute θ and ϕ for each camera and compute the pose density using kernel density estimation. When calculating this density, θ and ϕ are weighted similarly. Fig. 8 illustrates the visualization of θ and ϕ angles and their corresponding camera poses.

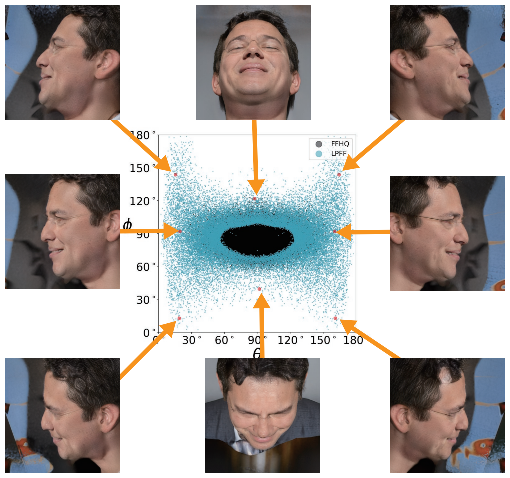

Figure 8: Visualization of camera poses.

1.8. EG3D Pipeline

We divide the EG3D model into three camera pose-dependent modules: Generator G, Renderer R, and Discriminator D, as shown in Fig. 11. The attribute correlations between pose and other semantic attributes in the dataset are faithfully modeled by feeding c_g into G. R and D are always fed with the same camera parameters, c_r. c_r help D ensure multi-view-consistent super resolution and direct R in how to render the final images from various camera views.

model	$c_g = c_{avg}$ $c_r \sim$ FFHQ	$c_g = c_{avg}$ $c_r \sim$ LPFF	$c_g \sim$ FFHQ $c_r \sim$FFHQ	$c_g \sim$ FFHQ $c_r \sim$LPFF	$c_g \sim$ LPFF $c_r \sim$FFHQ	$c_g \sim$ LPFF $c_r \sim$LPFF	$c_g \sim$ FFHQ $c_r = c_g$	$c_g \sim$ LPFF $c_r = c_g$
E^{FFHQ}_{var1}	6.523	23.598	4.273	22.318	23.698	36.641	4.025	23.301
$E^{FFHQ}_{var1-fixed}$	**7.689**	23.962	**6.572**	22.537	22.567	33.063	6.102	25.115
E^{Ours}_{var1}	7.997	**20.896**	6.623	**19.738**	**21.300**	**22.074**	**6.093**	**16.026**

Table 5: FID (↓) for EG3D generators that are trained on different datasets. We fixed the buggy tri-plane in E^{FFHQ}_{var1}, and retrained the EG3D model using the (XY, XZ, ZY) plane on the *FFHQ* dataset. The obtained model is named as $E^{FFHQ}_{var1-fixed}$.

1.9. Image Samples from *LPFF* Dataset

As shown in Fig. 9, we present the image samples from our *LPFF* dataset. We used the official API from Flickr to obtain image metadata (including URLs) by searching portrait-related keywords (e.g., "portrait", "people"), all images are under CC-BY-2.0[4], Public-Domain-Mark-1.0 [5], CC-BY-SA-2.0 [6], or CC0-1.0 [7] licenses. Because the *FFHQ* dataset includes metadata for its photos (including copyright information and URLs), we remove the raw image when the *FFHQ* metadata already includes the raw image's photo URL. The non-portrait images were removed by face detectors (Dlib and face alignment), while low-resolution images were filtered out automatically.

The images in the *LPFF* dataset are high-quality, large-pose, with variations on gender, age, race, expression, and lighting.

1.10. Additional StyleGAN2-ada Results

1.10.1 StyleGAN2-ada Inversion Results

We present StyleGAN models' large-pose data inversion results in Fig. 12 and 13. The testing images are collected from Unsplash and Pexels (out of the training datasets of all the models). The inversion is achieved by employing a 500-step latent code optimization to minimize the distance between the synthesis image and the target image. The optimization is performed in $W+$ latent space.

1.10.2 InterfaceGAN Pose Manipulation on StyleGAN Projection Results

After projecting the real large-pose images into the StyleGAN models' latent space, we utilize the yaw angle editing directions to edit the obtained latent codes (Fig. 14 and 15).

1.10.3 InterfaceGAN Attribute Manipulation on StyleGAN Projection Results

After projecting the real large-pose images into the models' latent space, we utilize the attribute editing directions to edit the obtained latent codes (Fig. 16 and 17).

We use the attribute classifiers provided by StyleGAN[1] to label the attribute scores of random latent codes, use InterfaceGAN to compute semantic boundaries for each model, and then use the boundaries to edit the projected latent codes.

1.11. Additional EG3D Results

1.11.1 EG3D Generation Results

We present the uncurated examples synthesized by models trained on our datasets in Fig. 18 and 19.

1.11.2 EG3D Extrapolation to Steep Camera Angles

Fig. 21 shows the comparison results of the steep angle generation results of the EG3D models trained on different datasets.

1.11.3 EG3D Image Inversion Results

Fig. 22 and Fig. 23 show the results of EG3D multi-view image inversion. We use 500-step latent code optimization to fit four testing images of a single identity from FaceScape [4]. Then we render the obtained latent codes from four novel views. The optimization is performed in $W+$ space, and the generators are conditioned on the average camera parameters.

Fig. 26 and 27 show the additional results of EG3D single-view image inversion. We perform PTI [2] inversion to single-view testing images collected from Unsplash and Pexels. The pivot latent codes are obtained by performing a 250-step latent code optimization in $W+$ space, and the models are fine-tuned using a 350-step PTI optimization.

References

[1] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 4401–4410, 2019. 4

[2] Daniel Roich, Ron Mokady, Amit H. Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.*, 42(1), 2022. 2, 4

[4]https://creativecommons.org/licenses/by/2.0/

[5]https://creativecommons.org/publicdomain/mark/1.0/

[6]https://creativecommons.org/licenses/by-sa/2.0/

[7]https://creativecommons.org/publicdomain/zero/1.0/

[3] WuJie1010. Facial-expression-recognition.pytorch. https://github.com/WuJie1010/Facial-Expression-Recognition.Pytorch. 2

[4] Haotian Yang, Hao Zhu, Yanru Wang, Mingkai Huang, Qiu Shen, Ruigang Yang, and Xun Cao. Facescape: A large-scale high quality 3d face dataset and detailed riggable 3d face prediction. In *IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 598–607, 2020. 4

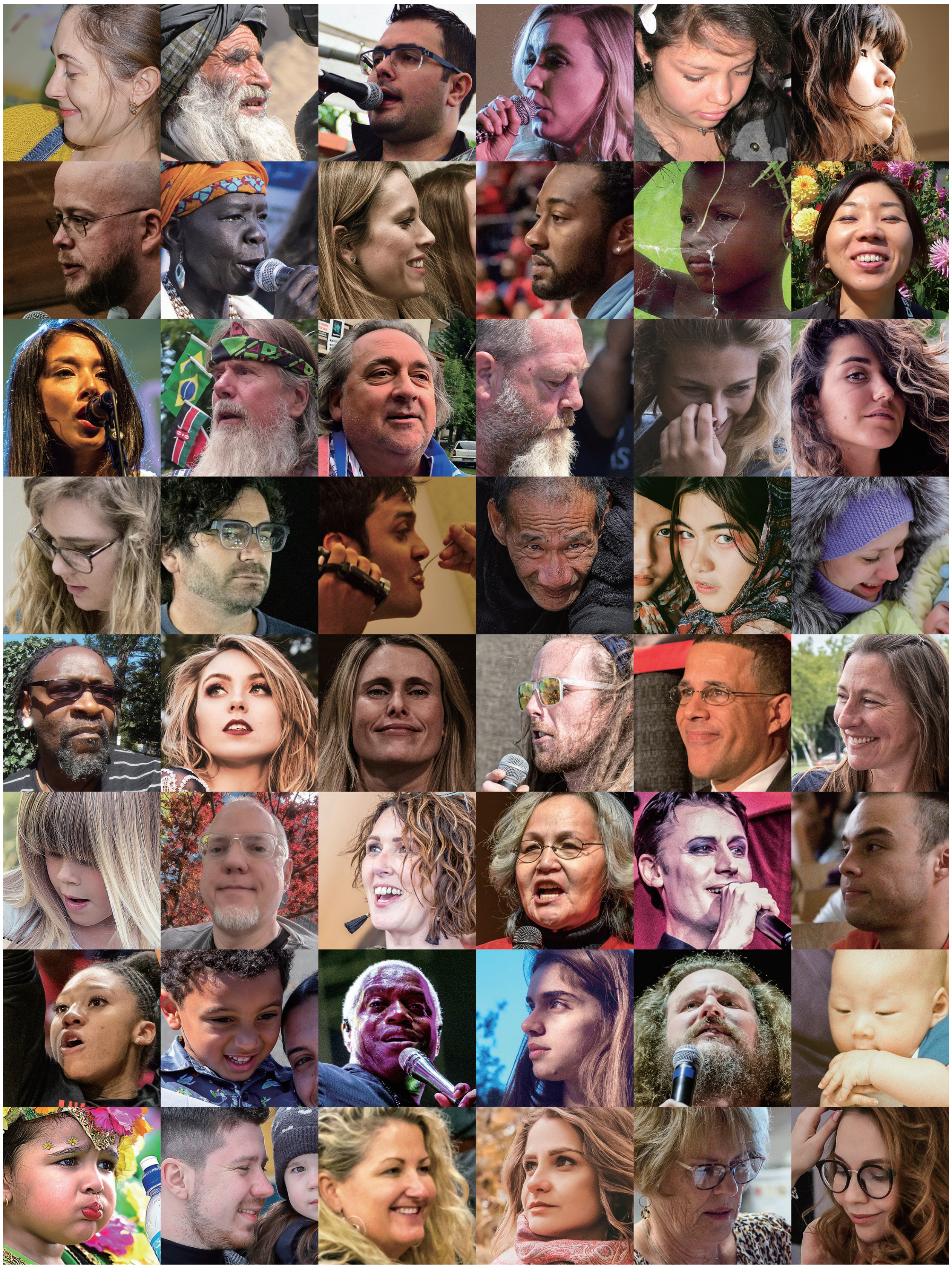

Figure 9: Random image samples from the *LPFF* dataset. The presented images are aligned using the EG3D image alignment functions, at 512^2 resolution.

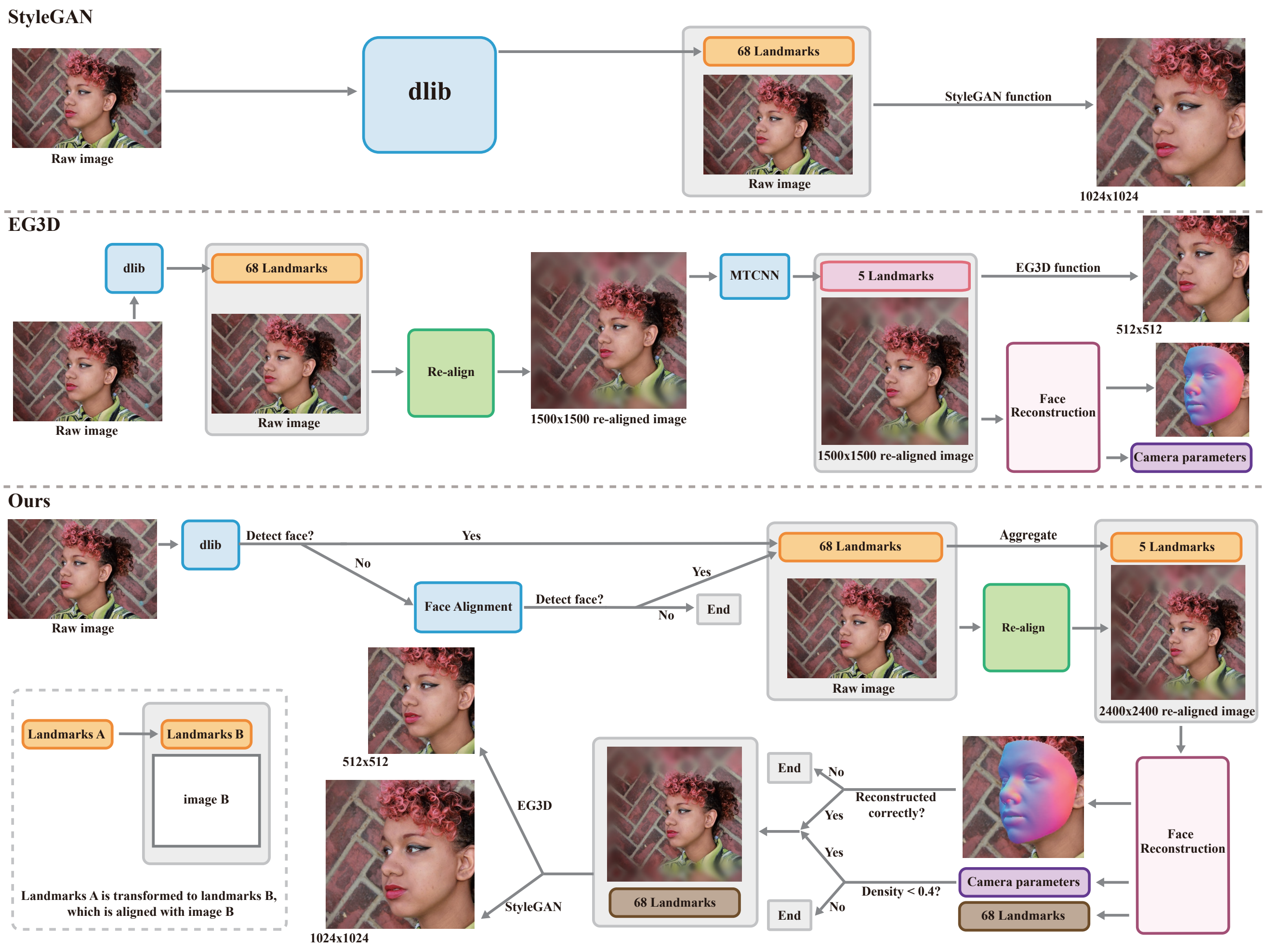

Figure 10: Image processing pipelines of StyleGAN, EG3D, and ours.

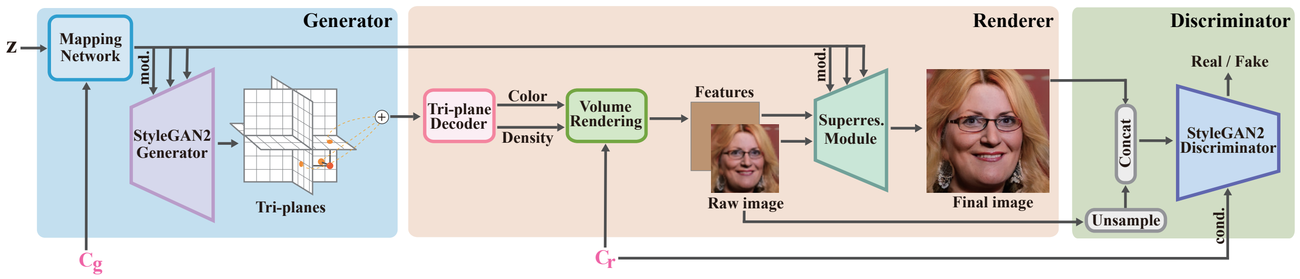

Figure 11: We split the EG3D model into three modules: Generator, Renderer, and Discriminator. We define the two kinds of camera parameters that are inputted into the EG3D modules as: c_g fed into the Generator is used to faithfully model the attribute correlations and will influence the face geometry and appearance. c_r guides the Renderer and the Discriminator to render the final images from different camera views while ensuring multi-view-consistent super resolution.

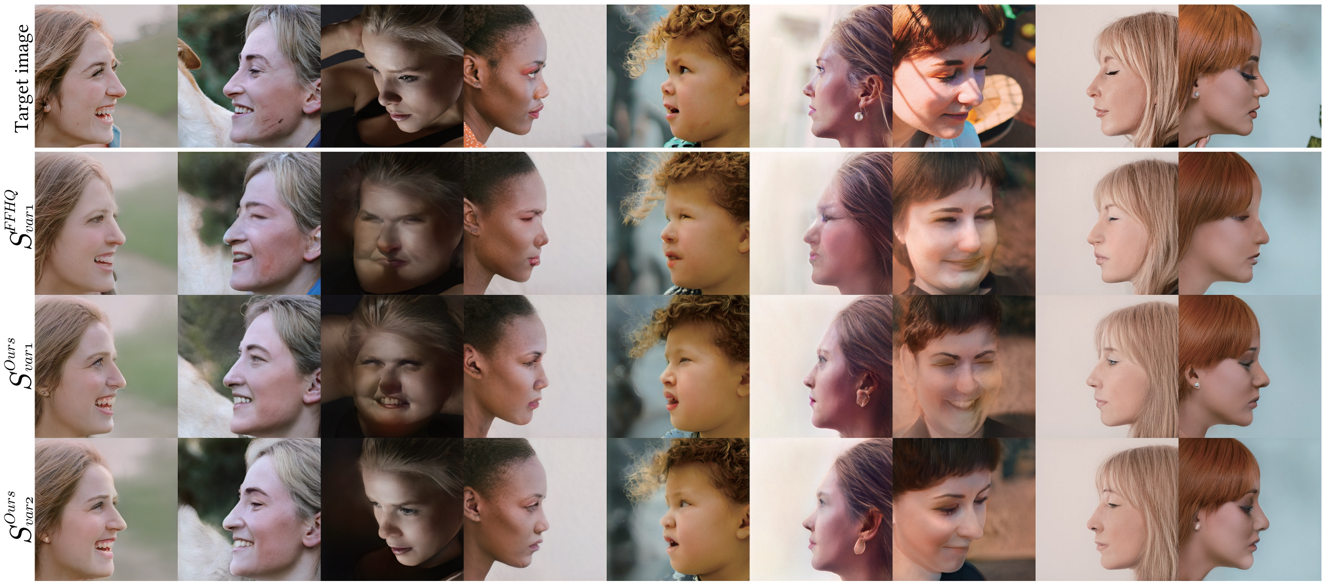

Figure 12: More qualitative comparison results for StyleGAN inversion.

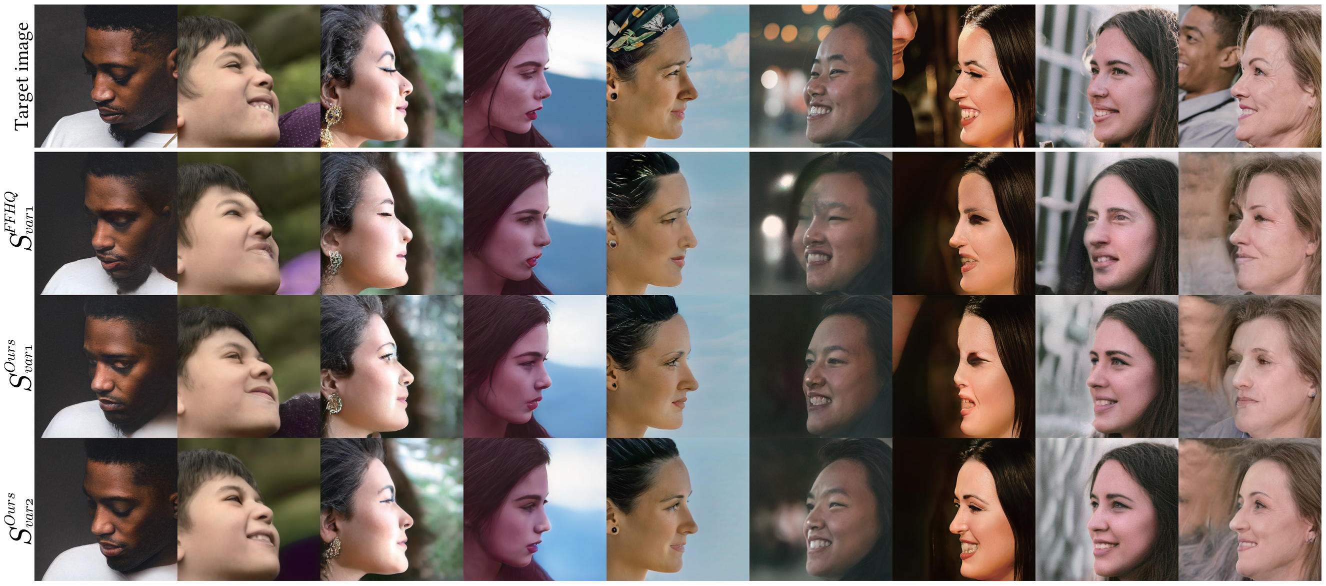

Figure 13: More qualitative comparison results for StyleGAN projection.

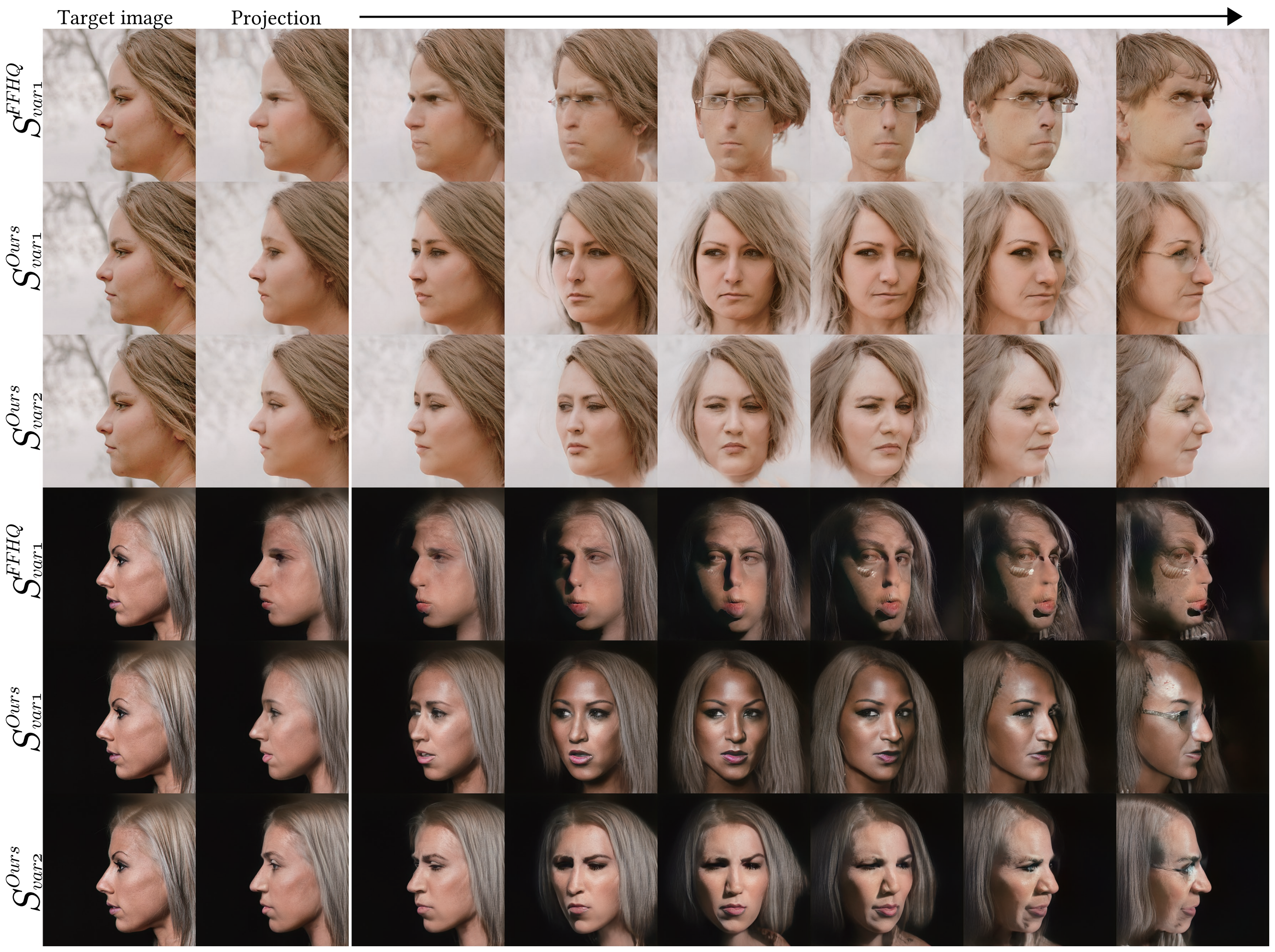

Figure 14: More qualitative comparison results for InterfaceGAN pose manipulation on StyleGAN inversion results.

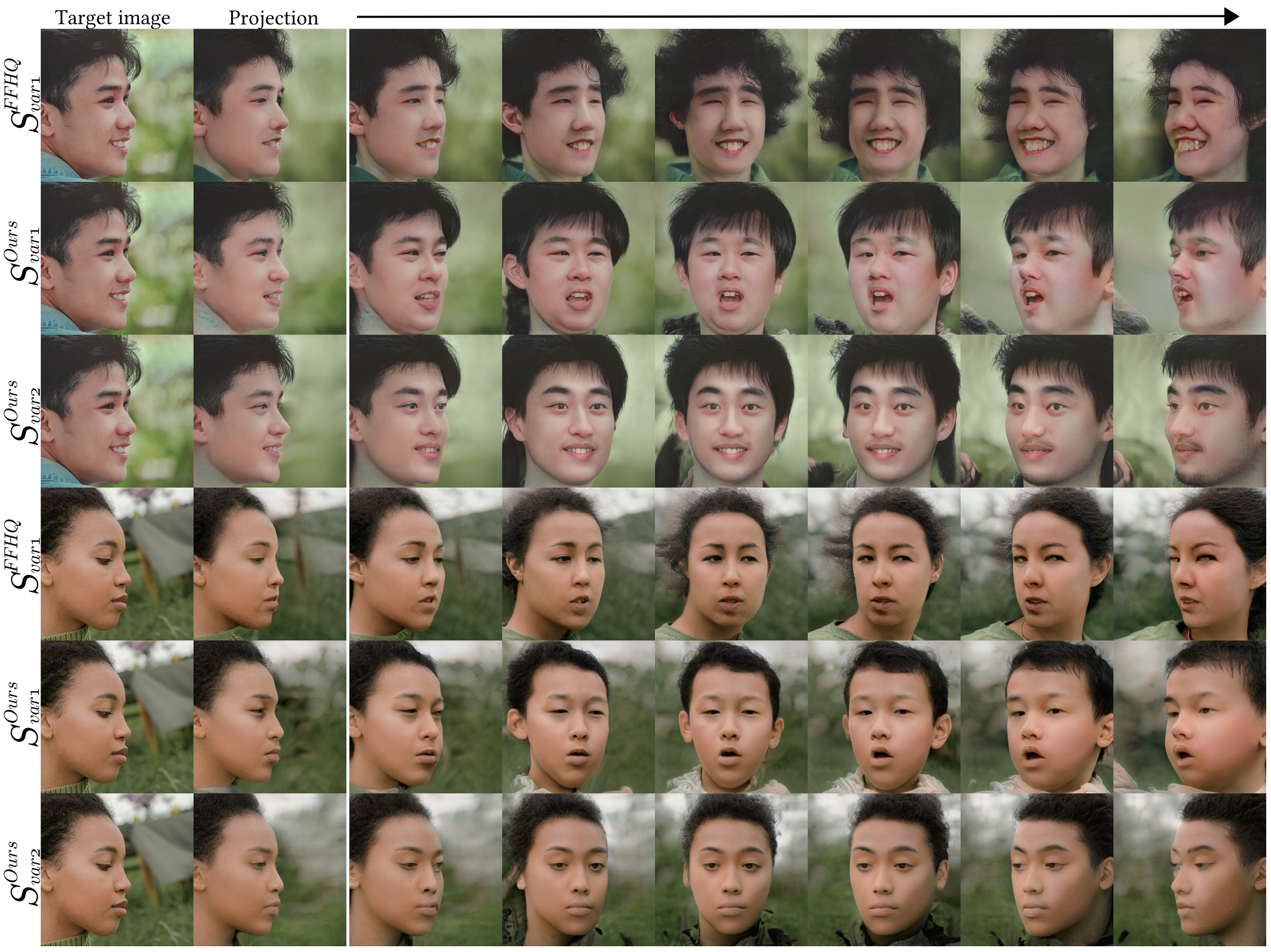

Figure 15: More qualitative comparison results for InterfaceGAN pose manipulation on StyleGAN inversion results.

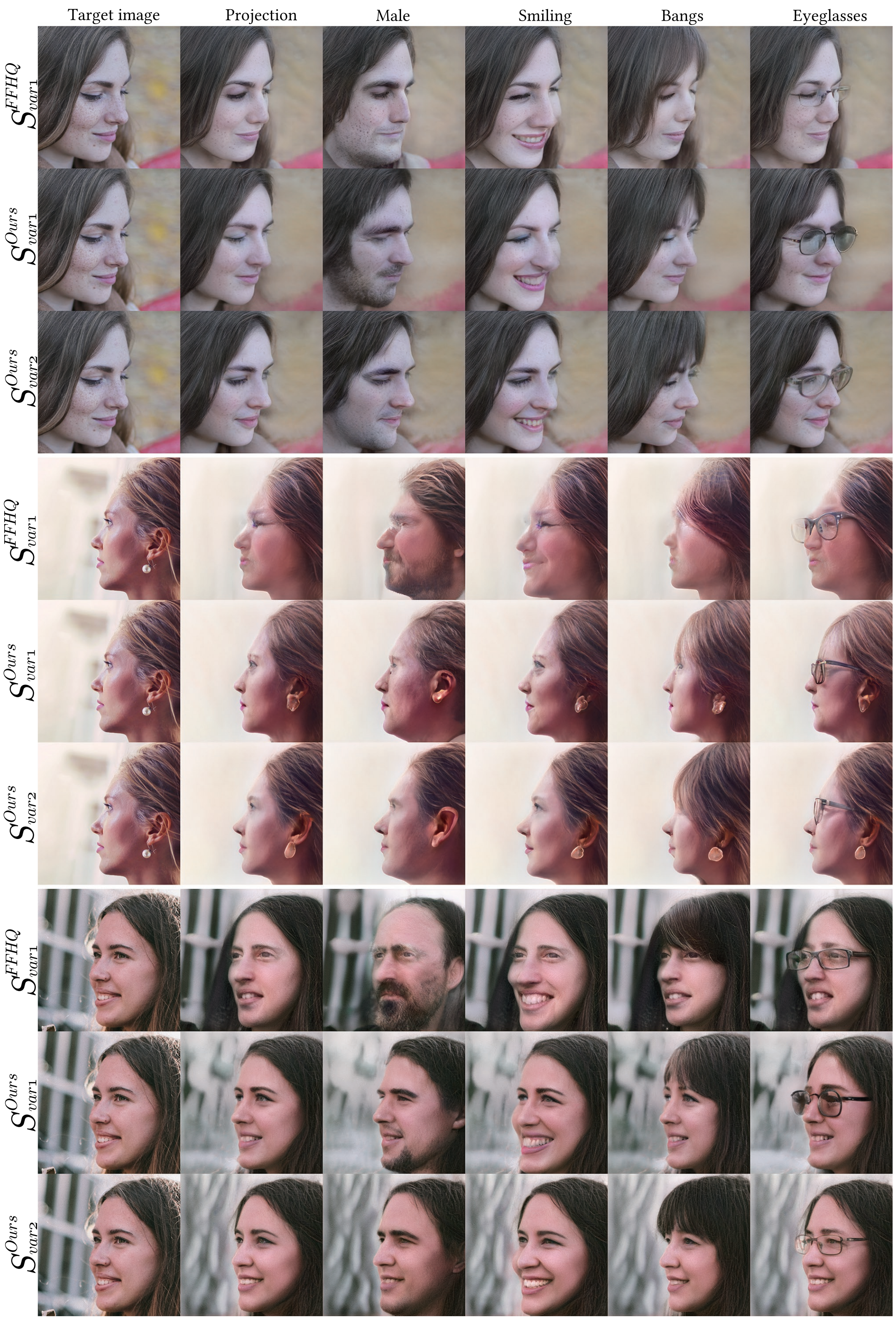

Figure 16: More qualitative comparison results for InterfaceGAN semantic attribute manipulation on StyleGAN inversion results.

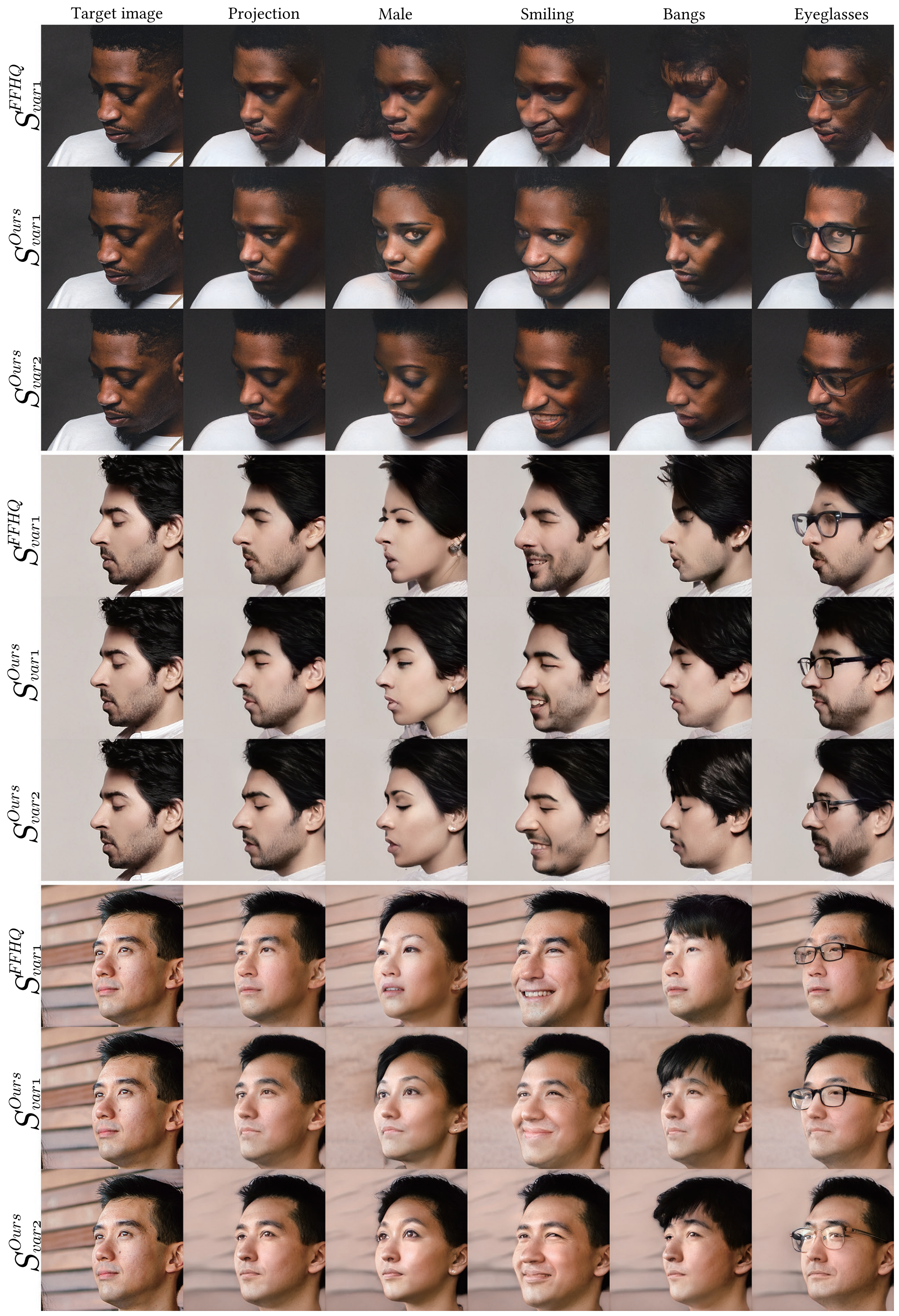

Figure 17: More qualitative comparison results for InterfaceGAN semantic attribute manipulation on StyleGAN inversion results.

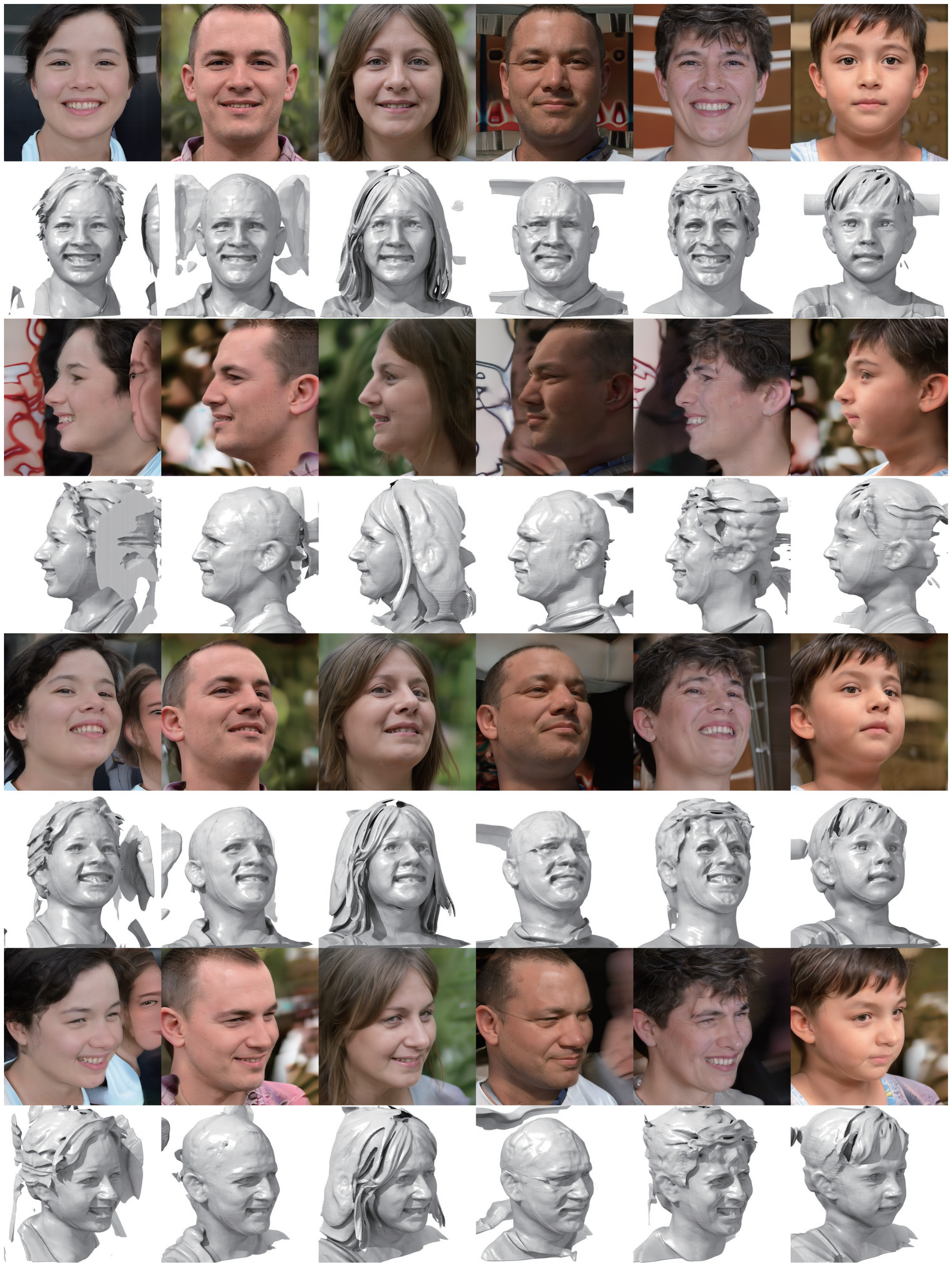

Figure 18: Uncurated examples synthesized by E_{var1}^{Ours}, with truncation ($\psi = 0.6$).

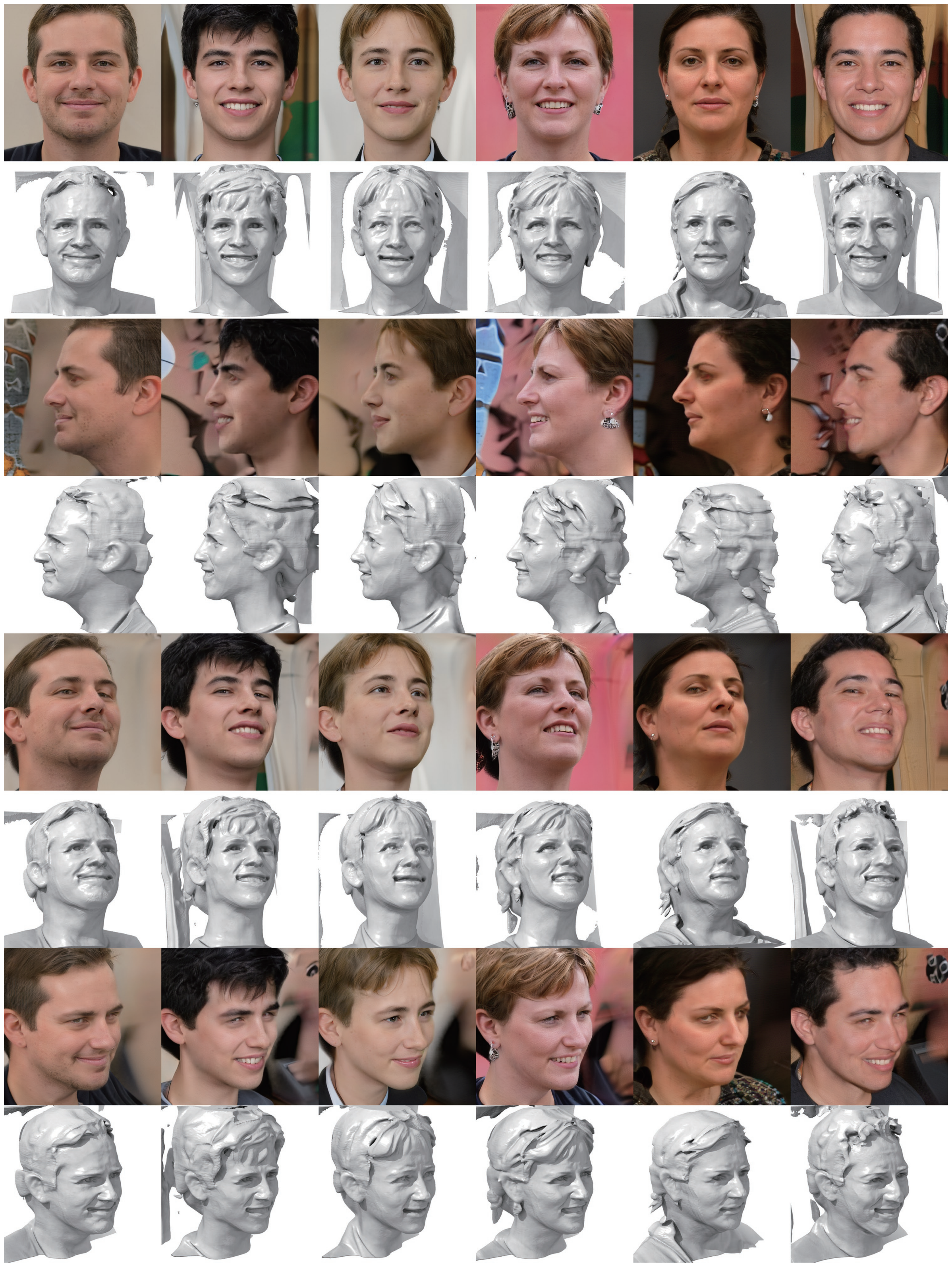

Figure 19: Uncurated examples synthesized by E_{var2}^{Ours}, with truncation ($\psi = 0.6$).

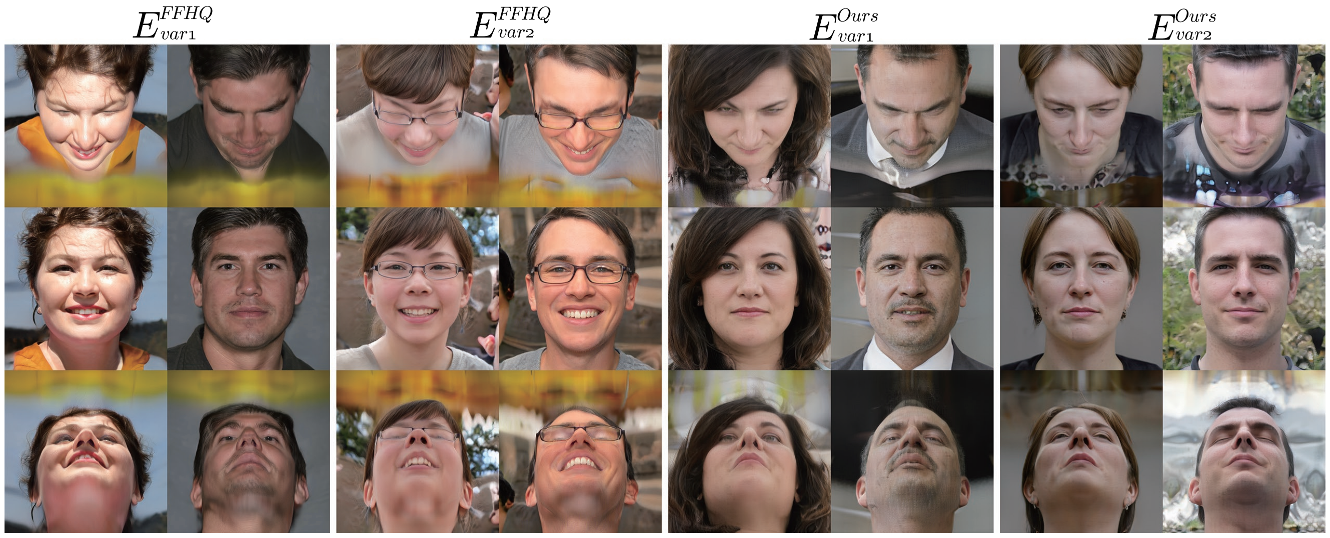

Figure 20: Extrapolation to steep pitch angles, with truncation ($\psi = 0.7$).

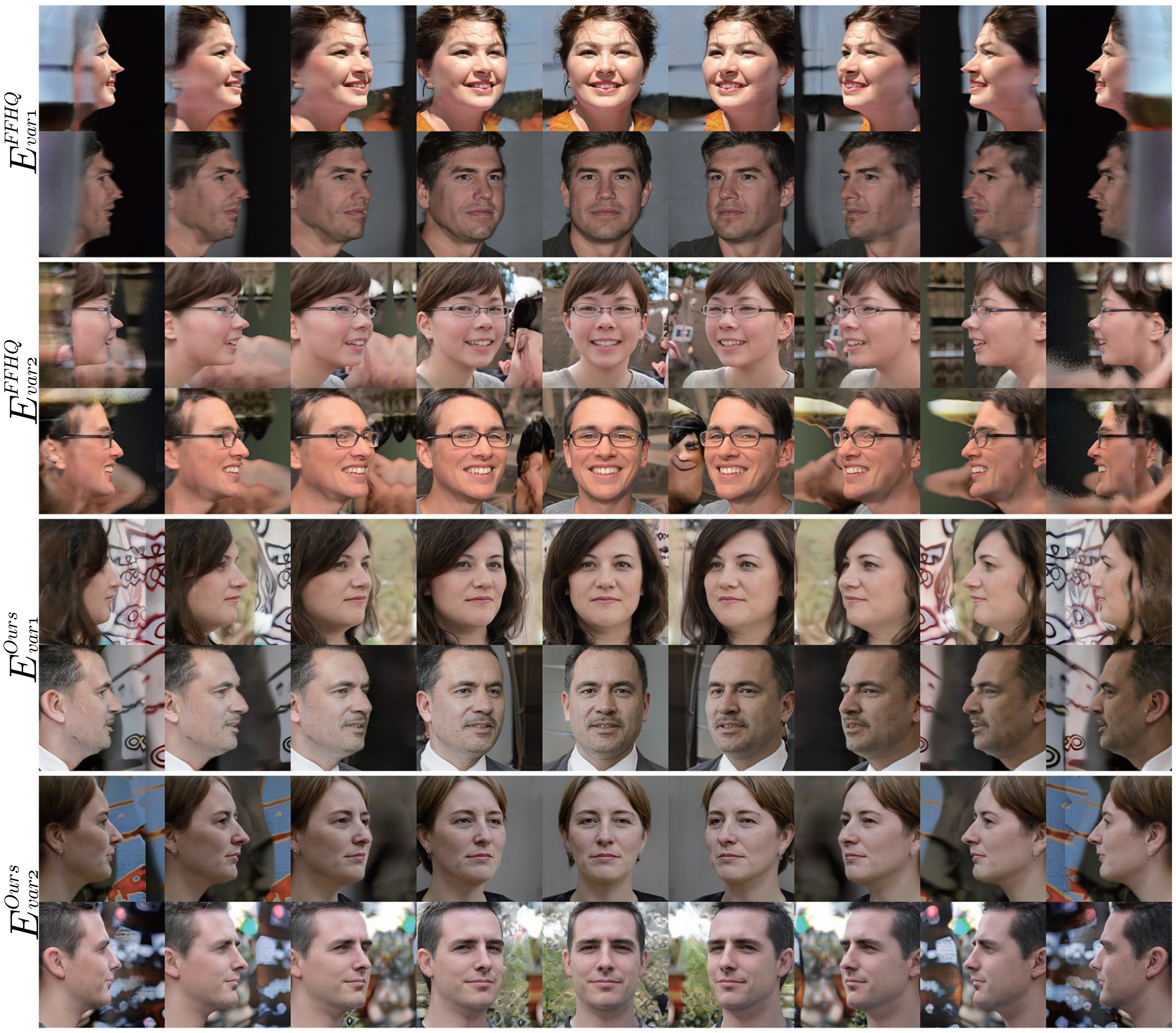

Figure 21: Extrapolation to steep yaw angles, with truncation ($\psi = 0.7$).

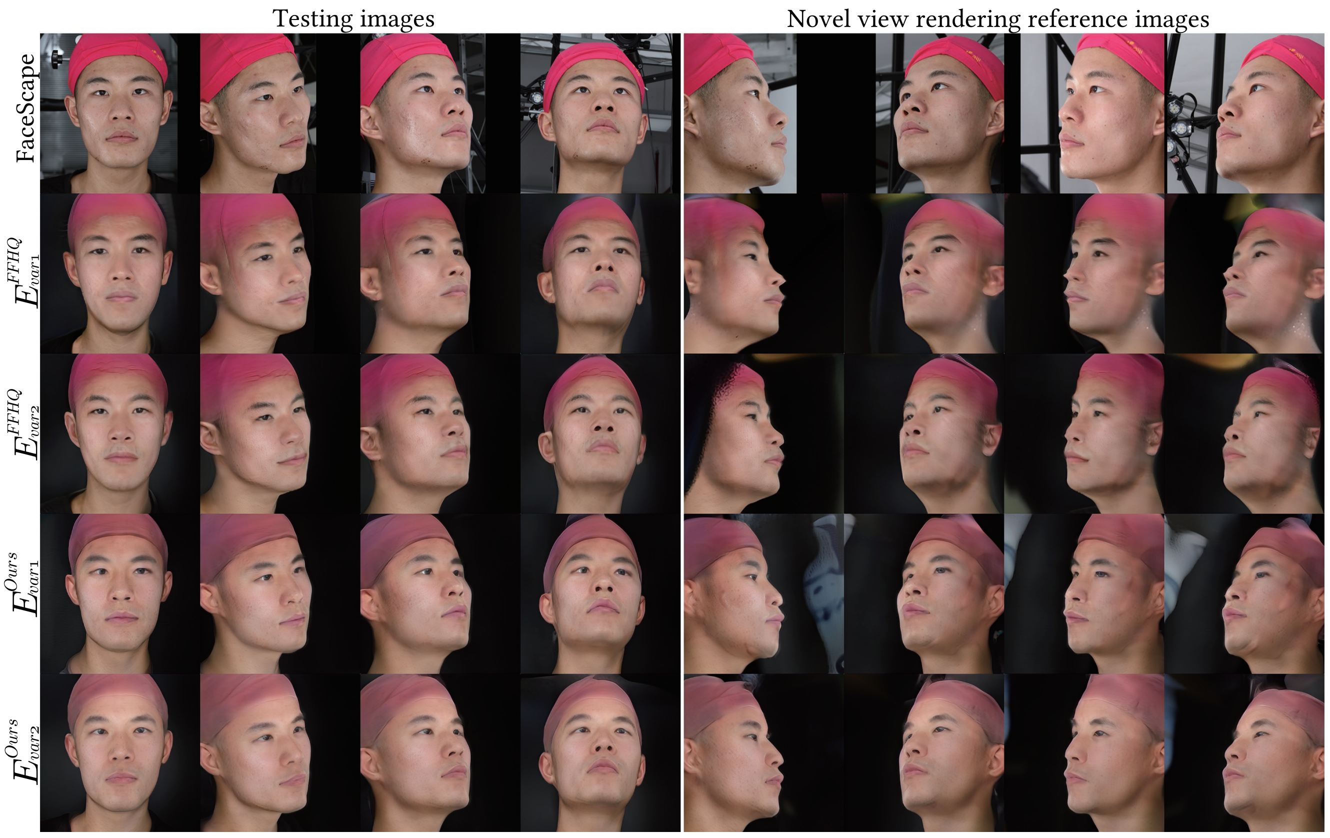

Figure 22: More EG3D multi-view image inversion results.

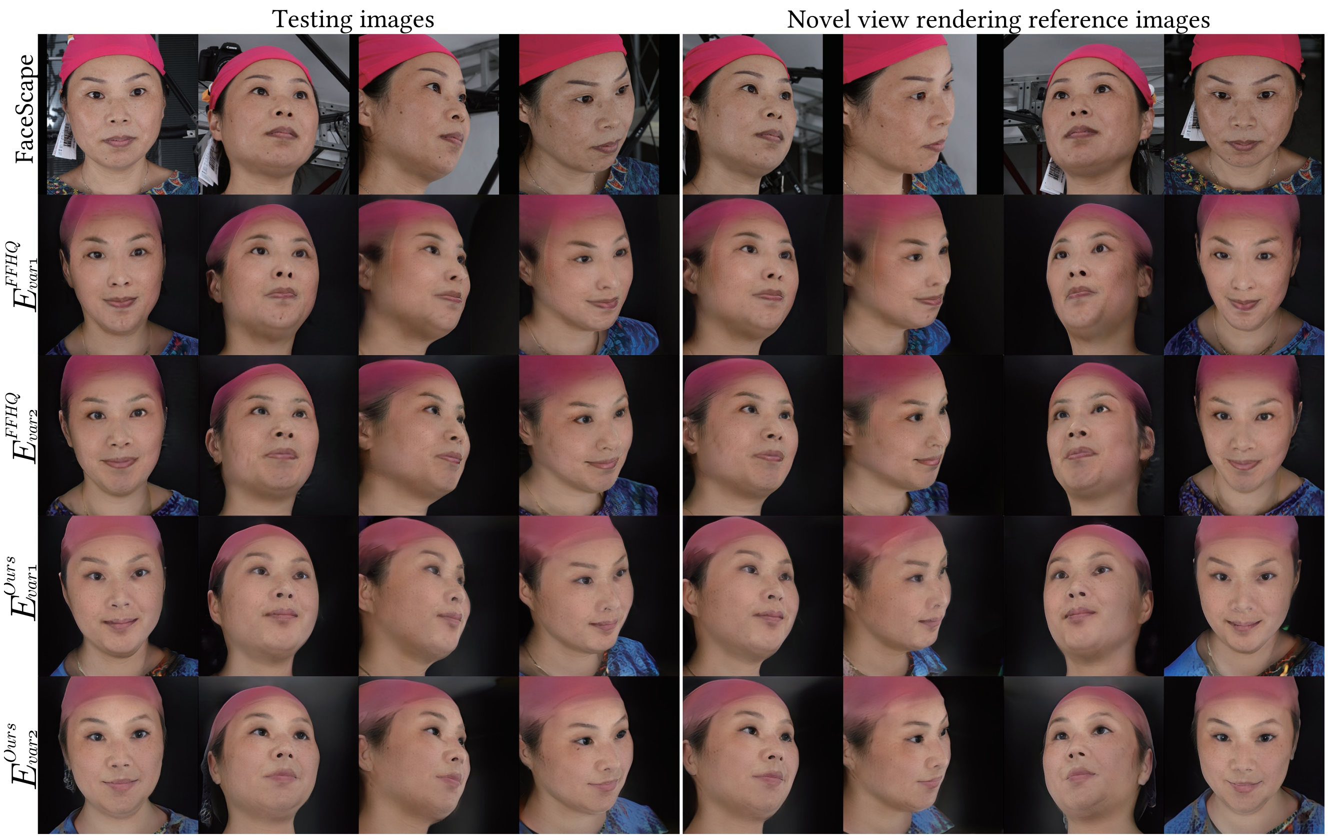

Figure 23: More EG3D multi-view image inversion results.

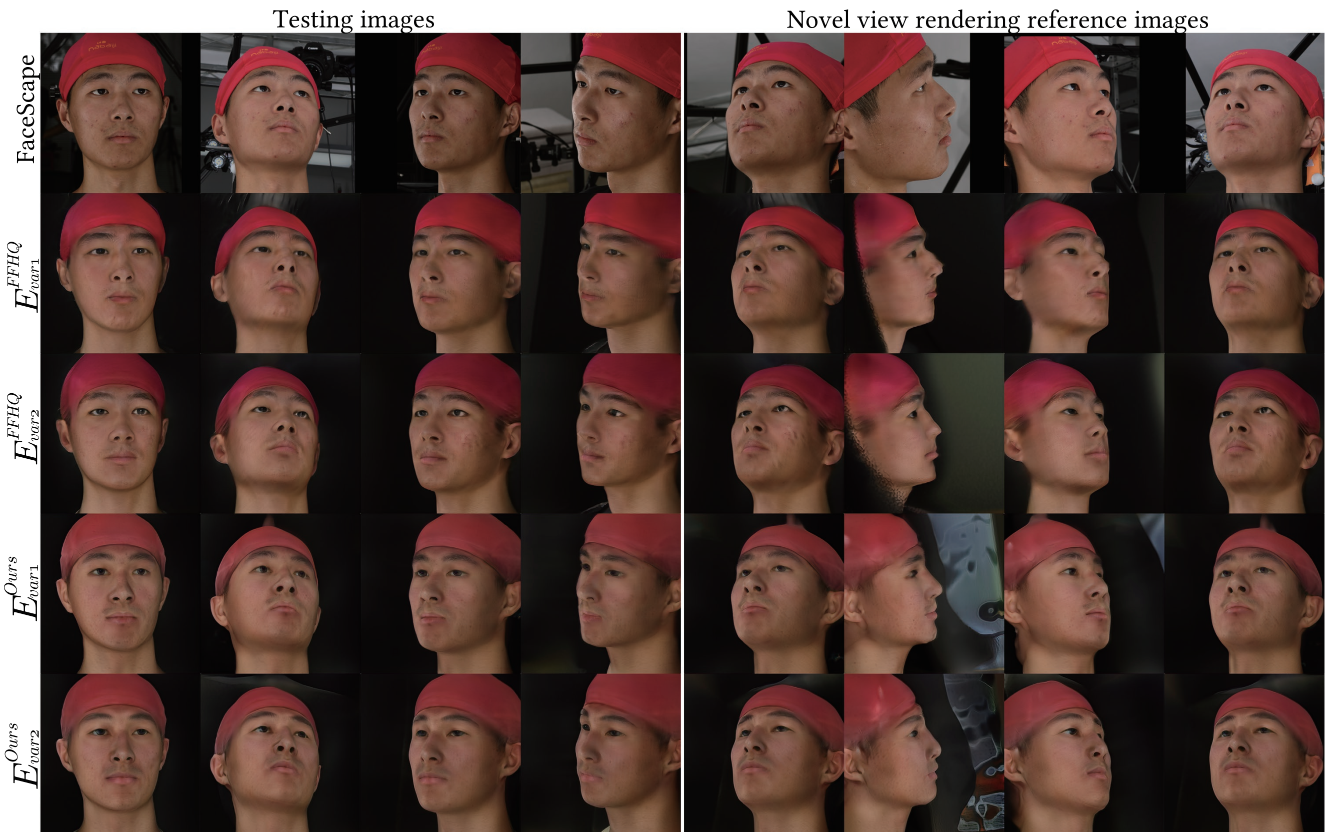

Figure 24: More EG3D multi-view image inversion results.

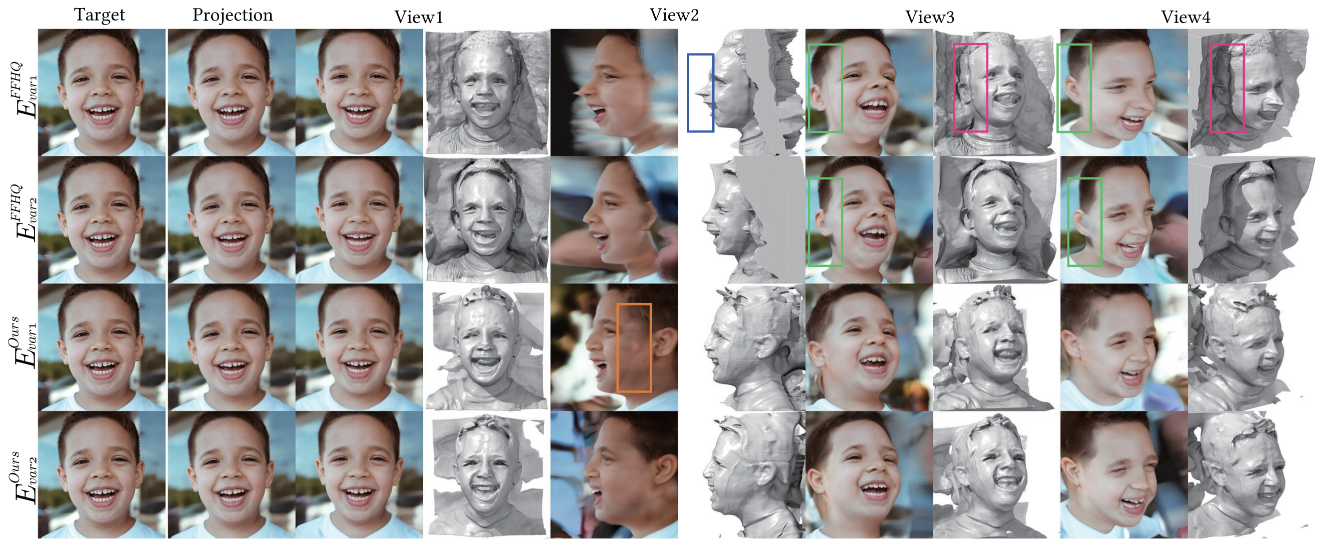

Figure 25: More EG3D single-view image inversion results. Due to the adhesion between the head and the background in E_{var1}^{FFHQ} and E_{var2}^{FFHQ}, ears are missing in View 2, and there are distortions in the ears and neck in Views 3 and 4 (highlighted by green boxes). E_{var1}^{FFHQ} also exhibits a pointed nose (highlighted by a blue box) and "seam" artifacts (highlighted by pink boxes). Additionally, compared to E_{var2}^{Ours}, there are some blurry skin artifacts present in E_{var1}^{Ours} (highlighted by an orange box).

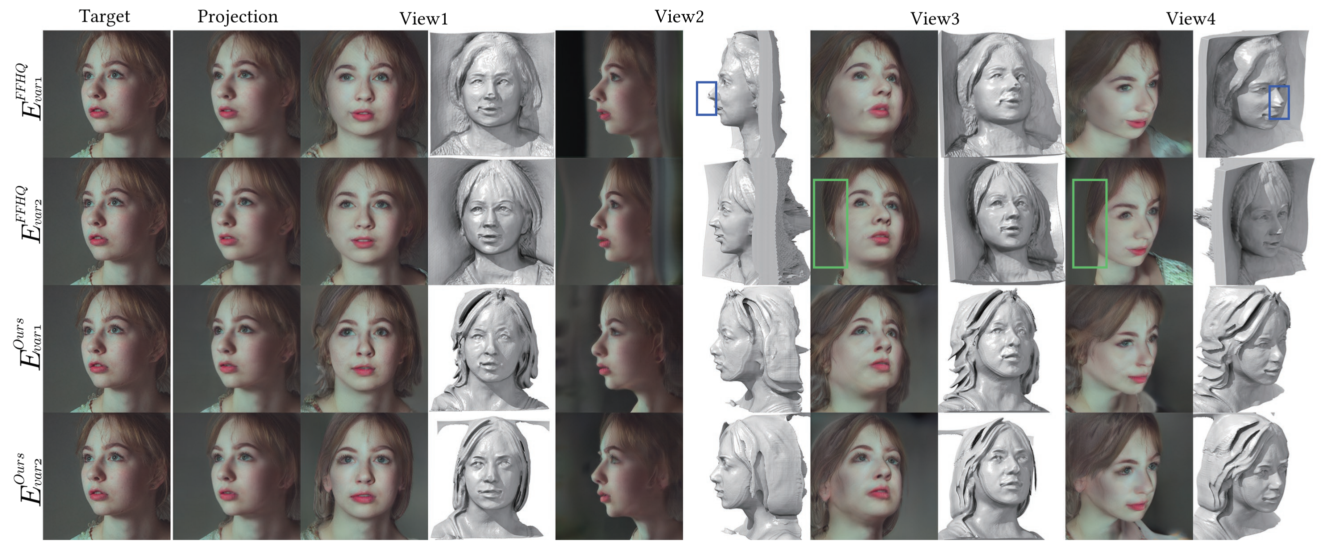

Figure 26: More EG3D single-view image inversion results. E_{var1}^{FFHQ} has a hole on the nose (highlighted by blue boxes), while E_{var2}^{FFHQ} exhibits adhesion between the head and the background (adhesion by green boxes).

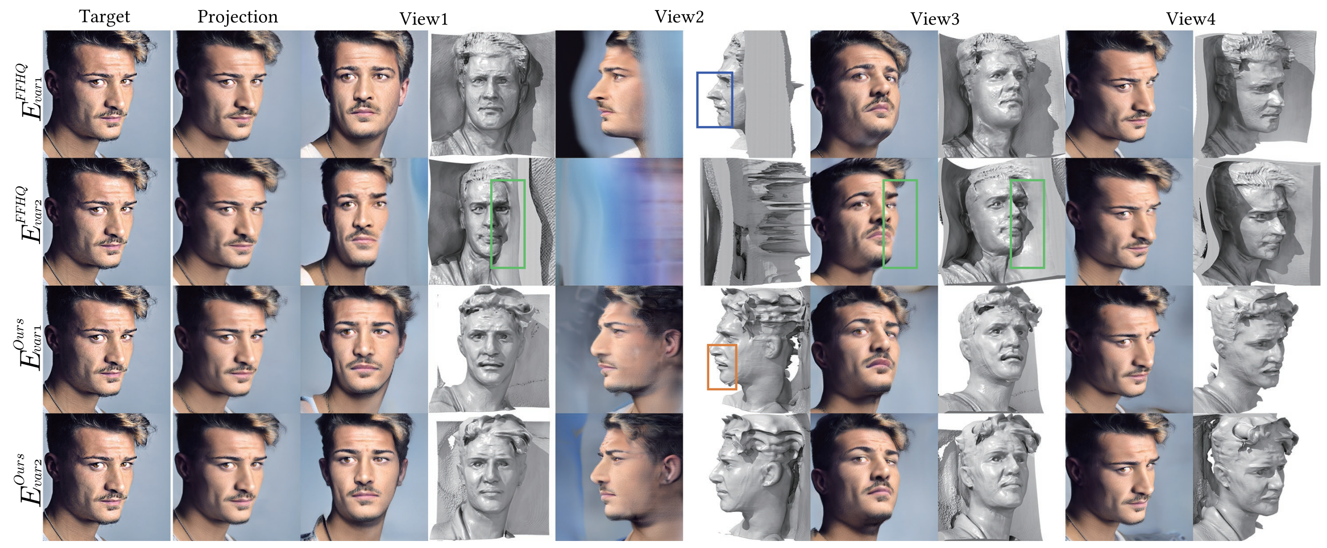

Figure 27: More EG3D single-view image inversion results. E_{var1}^{FFHQ} has a pointed nose (highlighted by a blue box). A severe distortion of background exists in E_{var2}^{FFHQ} (highlighted by green boxes). E_{var1}^{Ours} exhibits unnatural lips (highlighted by an orange box).

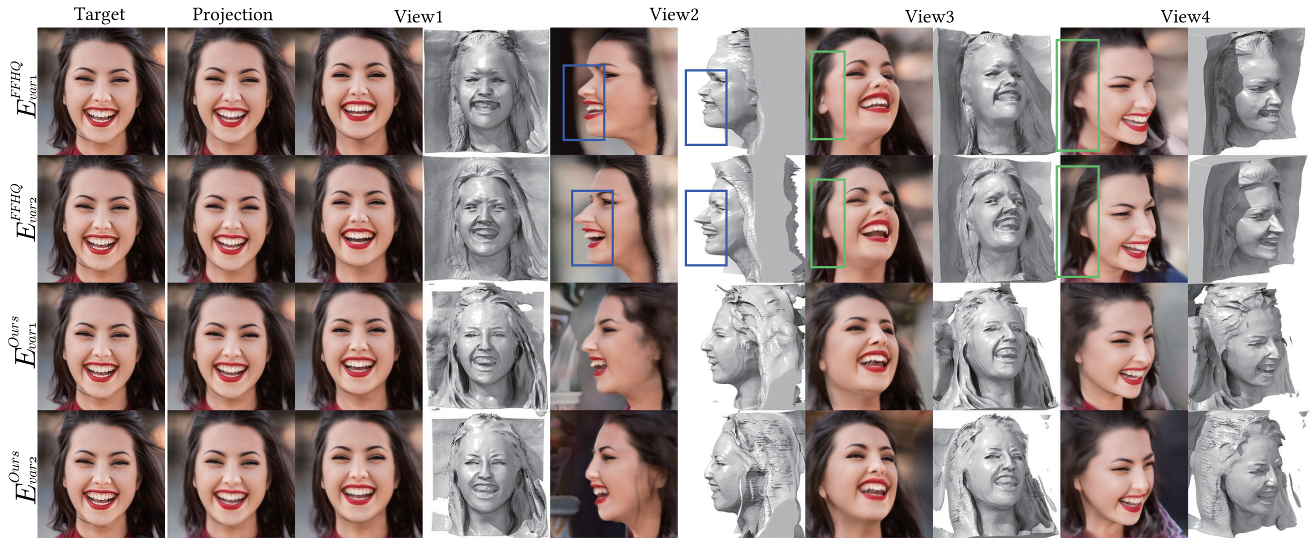

Figure 28: More EG3D single-view image inversion results. The results for E_{var1}^{FFHQ} and E_{var2}^{FFHQ} exhibit distorted faces (highlighted by blue boxes) and distorted ears and necks (highlighted by green boxes).