3D-aware Blending with Generative NeRFs -Supplementary Material-

Hyunsu Kim [1] Gayoung Lee [1] Yunjey Choi [1] Jin-Hwa Kim [1,2] Jun-Yan Zhu[3]

[1]NAVER AI Lab [2]SNU AIIS [3]CMU

Overview of the supplementary material.

- Please refer to the code, data, and results on our website: https://blandocs.github.io/blendnerf.
- Experimental details are described in Section A.
- Details of inversion are described in Section B.
- Details of local alignment are described in Section C with an extra user study.
- Details of 3D-aware blending are described in Section D.
- 3D-aware blending in StyleSDF is described in Section E. Our method can be applied to Signed Distance Fields (SDF) beyond NeRFs.
- Details of user studies are described in Section F.
- Failure cases are described in Section G.
- Societal impact is discussed in Section H.
- Additional qualitative results are in Section I.

A. Experimental details

Baselines.

- **Poisson Blending** [24] is implemented in OpenCV [4]. We use `cv2.seamlessClone` in the `cv2.NORMAL_CLONE` cloning type.
- **StyleGAN3**: As there is no official projection code in StyleGAN3 [16], we use unofficial implementation[1]. We follow the hyperparameters of StyleGAN2 [18] official projection algorithm[2] and set the number of optimization iterations as 1,000.
- **StyleMapGAN** [19] introduce *stylemap*, which has spatial dimensions in the latent space. We use the official pretrained networks with 8×8 and 16×16 *stylemap* for AFHQ [9] and 32×32 *stylemap* for FFHQ [17].
- **SDEdit** [22] transforms a noise-added image into a realistic image through iterative denoising. The total denoising step is a sensitive hyperparameter that decides blending quality. If it is too small, blending results are faithful to the input images but less realistic. If it is too large, blending results are less faithful to the input images but more realistic. We carefully select the number of iterations for the best quality; 300 for the small editing parts (eyes, nose, and lip) and 500 for the large editing parts (face and hair). To exploit the FFHQ-pretrained model [2][3], we implement guided-diffusion [11] version of SDEdit.
- **Latent Composition** [5] requires a mask to decide which area preserve. In our blending experiment, we need to preserve both the original and reference image, so we use a mask for the entire image.

Datasets. We select FFHQ [17] and AFHQv2-Cat [9] for our comparison experiments. FFHQ has 1024×1024 images, and AFHQ has 512×512 images. In SDEdit [22, 2], the pretrained model is trained on 256×256 FFHQ. Our backbone network EG3D [6] is trained on 512×512 FFHQ. In Table 1 of the paper, we upsample the blending results of SDEdit and EG3D using bilinear interpolation for a fair comparison. Other baselines in FFHQ and all methods in AFHQ have the same resolution with the corresponding datasets. As the StyleMapGAN network is trained on AFHQv1, we fine-tune the networks using AFHQv2-Cat. As EG3D uses a different crop version of FFHQ compared to the original FFHQ, we fine-tune the EG3D using the original crop version of FFHQ. We use ShapeNet-Car [7] (128×128) to further demonstrate the effectiveness of our method.

[1]https://github.com/PDillis/stylegan3-fun

[2]https://github.com/NVlabs/stylegan2-ada-pytorch

[3]https://github.com/yandex-research/ddpm-segmentation

Metrics. In Tables 1 and 2 of the main paper, we use the masked LPIPS (LPIPS$_m$) [30] to evaluate the faithfulness to the reference object. It needs a reference image to compute the score, and we use aligned reference images as pseudo-ground-truth images in both experiments: without and with 3D-aware alignment. Additionally, we always apply the alignment to our method in Tables 1 and 2 of the main paper.

Hyperparameters. In the 3D-aware blending, we optimize the latent code $\mathbf{w}_{\text{edit}} \in \mathbb{R}^{14\times512}$ of the $\mathcal{W}+$ space [1] for 200 iterations. The initial value of $\mathbf{w}_{\text{edit}}$ is the latent code of the original image $\mathbf{w}_{\text{ori}}$. Adam [20] optimizer is used with 0.02 learning rate, β_1 = 0.9, and β_2 = 0.999. We reformulate Eqn. 4 in the main paper to specify the hyperparameters as follows:

$$\begin{aligned}\mathcal{L}_{\text{image}} =&\|(\mathbf{1}-\mathbf{m})\circ \mathrm{I}_{\text{edit}} - (\mathbf{1}-\mathbf{m})\circ \mathrm{I}_{\text{ori}}\|_1 \\ &+ \lambda_1 \mathcal{L}_{\text{LPIPS}}((\mathbf{1}-\mathbf{m})\circ \mathrm{I}_{\text{edit}}, (\mathbf{1}-\mathbf{m})\circ \mathrm{I}_{\text{ori}}) \\ &+ \lambda_2\lambda_m \mathcal{L}_{\text{LPIPS}}(\mathbf{m}\circ \mathrm{I}_{\text{edit}}, \mathbf{m}\circ \mathrm{I}_{\text{ref}}),\end{aligned} \tag{1}$$

where $\lambda_1 = 1$, $\lambda_2 = 0.5$ for AFHQ, $\lambda_2 = 0.1$ for FFHQ except hair ($\lambda_2 = 0.3$). $\lambda_m = \frac{3\cdot H\cdot W}{\|\mathbf{m}\|_0}$ is a weighting parameter to give a high weight for the small target blending region $\mathbf{m}$; H and W denotes the height and width of the image, $\mathbf{m}$ is a binary mask $\mathbf{m} \in \{0,1\}^{3\times H\times W}$, and $||\cdot||_0$ is the L_0 norm that counts the total number of non-zero elements. Eqn. 5 in the paper is reformulated as follows:

$$\begin{aligned}\mathcal{L}_{\text{density}} =&\lambda_m \sum_{\boldsymbol{r}\in\mathcal{R}_{\text{ref}}}\sum_{\boldsymbol{x}\in\boldsymbol{r}} \|G_\sigma(\mathbf{w}_{\text{edit}};\boldsymbol{x}) - G_\sigma(\mathbf{w}_{\text{ref}};\boldsymbol{x})\|_1 \\ &+ \sum_{\boldsymbol{r}\in\mathcal{R}_{\text{ori}}}\sum_{\boldsymbol{x}\in\boldsymbol{r}} \|G_\sigma(\mathbf{w}_{\text{edit}};\boldsymbol{x}) - G_\sigma(\mathbf{w}_{\text{ori}};\boldsymbol{x})\|_1.\end{aligned} \tag{2}$$

The results of L1 loss in the image- and density-blending are normalized by the number of elements. We set λ in Eqn. 6 in the paper to 10.

B. Inversion details

We train our encoder to predict the camera pose of an image. It has a similar structure to the EG3D discriminator [6] but does not include minibatch discrimination [26] and involves adjustment of the number of input and output channels. Given that training images for generative NeRFs [6, 23] are pre-aligned with respect to scale and translation, the camera pose of the input image can be simplified as a rotation matrix; EG3D also samples camera poses on the surface of a sphere. As directly predicting the camera extrinsics $\in \mathbb{R}^{4\times4}$ is not easy, we convert the extrinsics to Euler angles $\in \mathbb{R}^3$. During inference, we transform it back to the camera extrinsics after obtaining the Euler angles from the encoder, which produces a more accurate pose estimation. Let the camera pose of the input image be $\mathbf{c}$.

We adopt Pivotal Tuning Inversion (PTI) [25] as our inversion method. In the first stage, we optimize the latent code $\mathbf{w} \in \mathcal{W}$ using the reconstruction loss $\mathcal{L}_{\text{rec}}$ as follows:

$$\mathcal{L}_{\text{rec}} = \|\mathrm{I} - G_{\text{RGB}}(\mathbf{w},\mathbf{c})\|_1 + \mathcal{L}_{\text{LPIPS}}(\mathrm{I}, G_{\text{RGB}}(\mathbf{w},\mathbf{c})), \tag{3}$$

where I is an input image and G_{RGB} is an image rendering function based on the generative NeRF G. $\|\cdot\|_1$ is the L1 distance and $\mathcal{L}_{\text{LPIPS}}$ is a learned perceptual image patch similarity (LPIPS) [30] loss. We optimize the latent code $\mathbf{w} \in \mathbb{R}^{512}$ for 300 iterations.

In the second stage, we fine-tune the generative NeRF G using the same reconstruction loss $\mathcal{L}_{\text{rec}}$ and an additional regularization loss $\mathcal{L}_{\text{reg}}$ as follows:

$$\begin{aligned}\mathcal{L}_{\text{reg}} =&\|G^f_{\text{RGB}}(\mathbf{w}_{\text{s}},\mathbf{c}) - G_{\text{RGB}}(\mathbf{w}_{\text{s}},\mathbf{c})\|_1 \\ &+ \mathcal{L}_{\text{LPIPS}}(G^f_{\text{RGB}}(\mathbf{w}_{\text{s}},\mathbf{c}), G_{\text{RGB}}(\mathbf{w}_{\text{s}},\mathbf{c})),\end{aligned} \tag{4}$$

where G^f_{RGB} represents the frozen version of G_{RGB}, and $\mathbf{w}_{\text{s}}$ is a randomly sampled latent code followed by linear interpolation with $\mathbf{w}$. The interpolation parameter is also randomly sampled in $[0, 1]$. The final loss function for optimizing G is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}, \tag{5}$$

where $\lambda_{\text{reg}} = 0.1$ and we optimize the G for 100 iterations.

Runtime. As described above, our inversion method consists of two-stage optimization. For a single image, the first stage (latent code) takes 23.6s, and the second stage (generative NeRF) takes 20.4s. We test the runtime on a single A100 GPU.

C. Local alignment

Local alignment is a fine-grained alignment between the target regions of two images. Even though we have matched two images through pose alignment, the scale and translation of target regions (*e.g.*, face, eyes, ears, *etc.*) might need to be further aligned, as the location and size of each object part differ across two object instances. Figure 1 illustrates our local alignment algorithm. In *Step 1*, we need to obtain 3D meshes M_{ori} and M_{ref} of input images using the Marching Cube algorithm [21] and the density fields. In *Step 2*, we first cast rays through the interior of the target region $\mathbf{m}$ and determine the intersected 3D points with the mesh M_{ori}. We define the set of points as a 3D point cloud P_{ori}. Similarly, we can get the reference's 3D point cloud P_{ref}.

Given two sets of 3D point clouds P_{ref} and P_{ori}, we use the Iterative Closest Point (ICP) algorithm [3, 8] to obtain

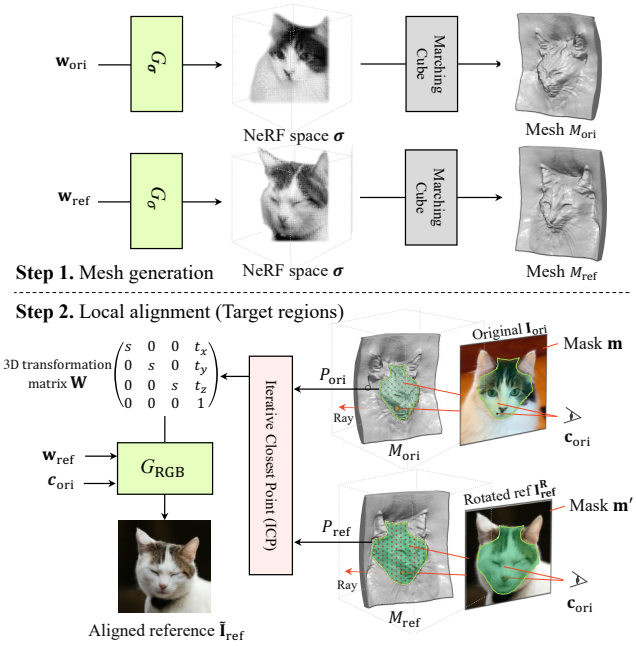

Figure 1: **Local alignment**: In **Step 1**, we first generate 3D meshes using the density field generator G_σ and the Marching Cube [21] algorithm. In **Step 2**, we calculate the intersected 3D points between the 2D mask $\mathbf{m}$ and the corresponding mesh M. Then, we use Iterative Closest Point (ICP) algorithm [3, 8] to estimate the 3D transformation matrix $\mathbf{W} \in \mathbb{R}^{4\times4}$ to align the 3D point clouds (P_{ori} and P_{ref}) in terms of the scale and the translation. Finally, we locally align the reference image $\tilde{\mathbf{I}}_{\text{ref}}$ by $G_{\text{RGB}}(\mathbf{w}_{\text{ref}}, \mathbf{c}_{\text{ori}}; \mathbf{W})$.

the 3D transformation matrix $\mathbf{W} \in \mathbb{R}^{4\times4}$. When we sample the 3D points to align the reference image, we transform the 3D point coordinates by multiplying the coordinate of sampled points with $\mathbf{W}$. We only use uniform scaling and translation in ICP, as we have already matched the rotation through *pose alignment* as described earlier in Section 3.1 of the main paper. Finally, we can generate the fully aligned reference image $\tilde{\mathbf{I}}_{\text{ref}}$ as follows:

$$\tilde{\mathbf{I}}_{\text{ref}} = G_{\text{RGB}}(\mathbf{w}_{\text{ref}}, \mathbf{c}_{\text{ori}}; \mathbf{W}). \tag{6}$$

Iterative Closest Point (ICP) [3, 8] is an algorithm to minimize the difference between two 3D point clouds. It is an iterative optimization process until we meet the threshold τ of difference or the maximum number of iterations $K = 20$. Please note that we do not use rotation in ICP as we align the rotation for entire objects in *pose alignment*, and refer to the exact details of ICP in our code.

3D transformation in the NeRF space. $\mathbf{W}$ is the 3D transformation matrix computed by the ICP algorithm. It transforms the point cloud of the blending region in the reference, aligning with the point cloud of the blending region in the original. A Neural Radiance Field learns a mapping

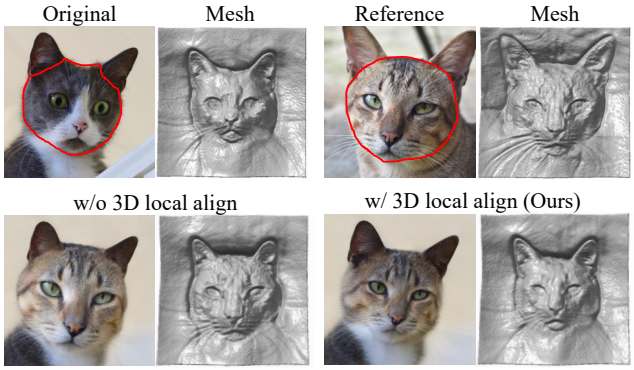

Figure 2: The effect of local alignment. Generative NeRF can align global object poses, but cannot handle differences in local parts. For example, in AFHQ-Cat, the proportions of a cat's face and ears vary. (Bottom left) It cannot be handled with pose estimation only, so local parts can be blended in the wrong places. (Bottom right) Our 3D local alignment method alleviates this issue and produces more natural results.

function f that outputs density σ and radiance $\mathbf{c}_{\text{RGB}}$ for a given 3D point $\boldsymbol{x} \in \mathbb{R}^3$; $f(\boldsymbol{x}) = (\sigma, \mathbf{c}_{\text{RGB}})$. 3D points are sampled along a ray, which depends on the camera pose. Let us assume that we define the matrix $\mathbf{W}$ as reducing scale. We cannot directly reduce the object scale in NeRF because the mapping function f is fixed. Instead, we can sample points more widely to reduce the object scale relatively. Renewed density σ' and radiance $\mathbf{c}'_{\text{RGB}}$ can be computed using the inverse matrix of $\mathbf{W}$ as follows $f(\mathbf{W}^{-1}\boldsymbol{x}) = (\sigma', \mathbf{c}'_{\text{RGB}})$.

Ablation study for local alignment. Pose alignment is a critical part, but it alone might not be enough to handle loosely aligned images such as AFHQv2-Cat. Figure 2 shows an ablation study of our 3D local alignment method. The cat in the original image has a smaller face than the reference cat. If we just apply pose alignment only, the blending result (bottom left in Figure 2) will create a much bigger face than humans normally expect. After locally aligning the face of the reference, we can obtain a natural-looking result with proper scale (bottom right in Figure 2). Figure 3 shows additional ablation results. We further examine local alignment by conducting a user study. In $\mathbf{60.3}\%$ of cases, users prefer our blending results with local alignment compared to those without alignment. Please see the details of the user study in Section F.

Runtime. Pose alignment takes 1.7s, and local alignment takes 4.2s, 4.6s, 5.9s for ears, eyes, face in AFHQv2-Cat [9]. The larger blending part takes more time to align. Our local alignment implementation uses the `Trimesh`[4] library, which provides pre-defined functions for triangular meshes

[4] https://trimsh.org/trimesh.html

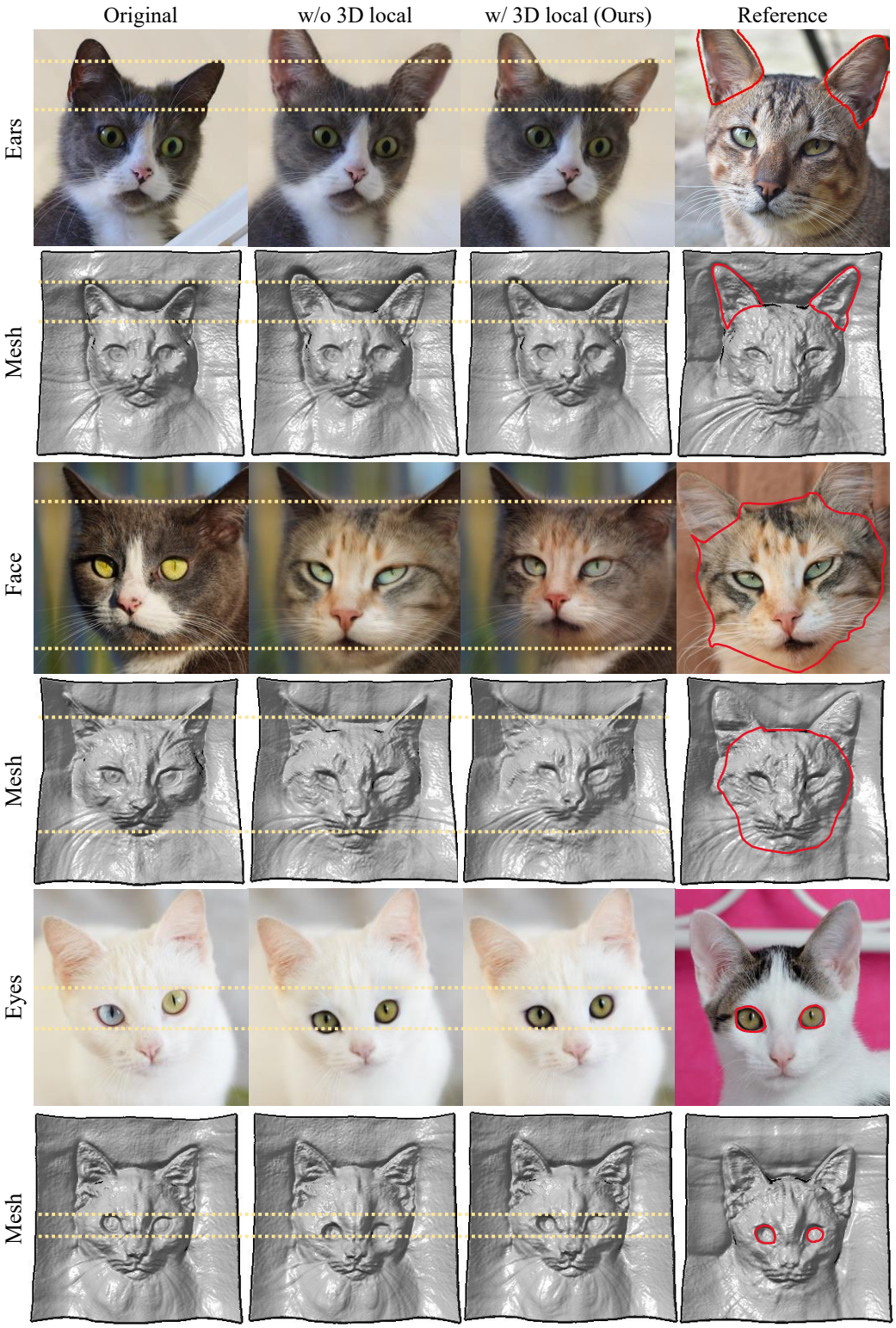

Figure 3: Ablation study of our 3D local alignment method in Section C. The leftmost and rightmost columns denote the original and reference images, respectively. Images in even rows are meshes corresponding to odd rows. Red lines denote the target blending mask, and yellow dotted lines are guidelines to be aware of alignment easily. Our 3D local alignment method (3rd column) shows more realistic blending results than those without local alignment (2nd column).

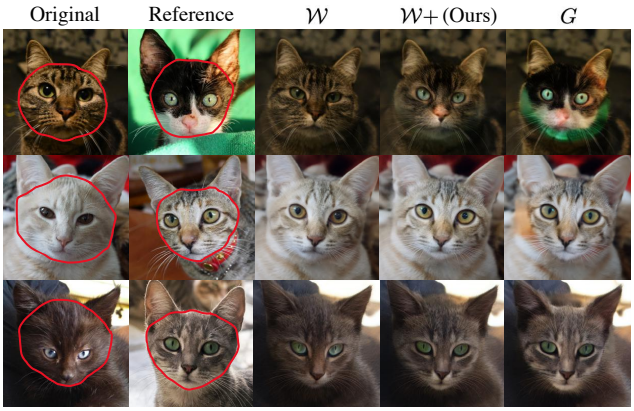

Figure 4: Blending comparison among the optimization spaces in Section D. The first and second columns denote original and reference images, respectively. The blending results of $\mathcal{W}$ space (3rd column) show realistic but less faithful to the reference images; see the details of eyes. The blending results of G space (rightmost column) show faithful but less realistic results. $\mathcal{W}+$ space (4th column) shows favorable results in both realism and faithfulness.

on the CPU. Implementing it with GPU-based libraries can further reduce alignment runtime.

D. Details of 3D-aware Blending

Choice for the optimization space. In our blending method, we optimize the latent code $\mathbf{w} \in R^{14\times512}$ in the extended latent space $\mathcal{W}+$ [1]. There are other options for optimization, such as an intermediate latent space $\mathcal{W}$ [17] or generator G. Figure 4 shows the comparison among optimization spaces in 3D-aware blending. $\mathcal{W}$ space shows realistic blending results but lacks faithfulness to the reference object. The blended image can not capture the green eyes of the reference object, as shown in the first row of Figure 4. Optimizing G shows faithfulness to the reference object but lacks realism. The boundaries of results look particularly odd. $\mathcal{W}+$ space shows great results in both realism and faithfulness, our method adopts to optimize the $\mathcal{W}+$ latent space.

Masks used in alignment and blending. At first, we apply *pose alignment* to rotate the reference to match the pose of the original. A user selects a mask $\mathbf{m}$ for the target blending region of the original image. The user selects another mask $\mathbf{m}'$ for the target blending region of the reference image, or it can be derived automatically using previous works [31, 13]. We blend images using the union of the masks, $\mathbf{m} \cup \mathbf{m}'$, and we redefine $\mathbf{m}$ as this union in Figure 4 in the main paper. Before blending in AFHQ, $\mathbf{m}$ and $\mathbf{m}'$ are used for *local alignment* to align target regions of two images, as shown in Figure 1. After local alignment, the target reference mask $\mathbf{m}'$ is also modified.

Figure 5: Our 3D-aware blending results of StyleSDF [23] in Section E. The third column is blending results in 1024^2 resolution. Our method can be applied to the SDF-based 3D-aware generator beyond NeRFs.

Runtime comparison with baselines. Poisson blending [24], StyleMapGAN [19], Latent Composition [5] take less than a second to blend images at 1024×1024 resolution, as they use low-level visual cues [24] or encoders [19, 5]. SDEdit [22] and StyleGAN3 [16] both require an iterative process. SDEdit takes 29.5s for 500 iterations (face and hair) and 20.0s for 300 iterations (nose, eyes, lip) at 256^2 image resolution. StyleGAN3 takes 69.8s for 512^2 and 106.5s for 1024^2 resolution. Our method takes 26.9s at 512^2 and we can further reduce the time to 12.1s by combining ours with Poisson blending (Section 3.3). All runtimes are measured in the same device with a single A100 GPU. Ours is faster than other optimization-based baselines but slower than the encoder-based methods. In the future, we can directly train an encoder using our image- and density-blending losses to reduce the runtime.

E. 3D-aware blending in StyleSDF

StyleSDF [23] generates high-fidelity view-consistent images in 1024×1024 resolution. The main difference between StyleSDF and EG3D [6] is StyleSDF uses Signed Distance Fields (SDF) as 3D representation. Besides, StyleSDF does not require camera pose labels to train the generator, unlike EG3D. In our 3D-aware blending method, we can use the SDF value d as a 3D signal, similar to the density σ in EG3D. If we assume a non-hollow surface, the SDF value can be

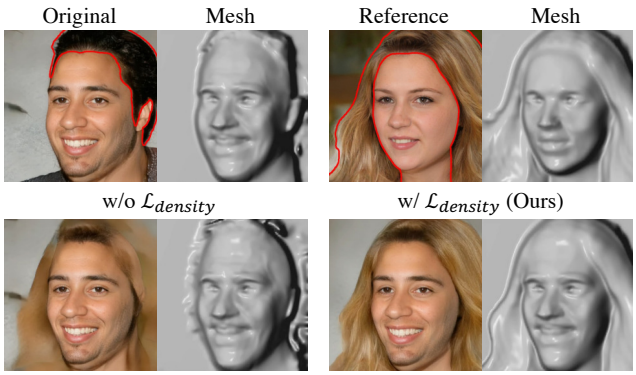

Figure 6: Ablation study of density-blending loss $\mathcal{L}_{\text{density}}$ in StyleSDF (Section E). This experiment uses SDF instead of volume density σ. Without $\mathcal{L}_{\text{density}}$, the blending result shows a blurry image, and the corresponding mesh can not reflect the geometry of the hair of the reference image. 3D signals such as density and SDF are key components in the 3D-aware blending method.

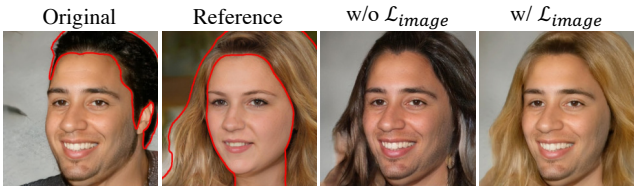

Figure 7: Ablation study of image-blending loss $\mathcal{L}_{\text{image}}$ in StyleSDF (Section E). We can control the degree to which the color of the reference object is reflected in the blended image, by adjusting the weight of the image-blending loss with respect to the reference image.

converted to the density σ as follows:

$$\sigma(\boldsymbol{x}) = \frac{1}{\alpha} \cdot \text{Sigmoid}(\frac{-d(\boldsymbol{x})}{\alpha}), \tag{7}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ is a 3D location and α is a learned parameter about the tightness of the density around the surface boundary. We use the same blending loss functions used in EG3D, except we replace the density σ with SDF d.

We demonstrate that our method can be applied to other 3D-aware generative models beyond EG3D. Figure 5 shows our 3D-aware blending results in StyleSDF using generated images. Figure 6 and Figure 7 show ablation studies of our blending loss terms: $\mathcal{L}_{\text{density}}$ and $\mathcal{L}_{\text{image}}$, respectively. The ablation studies show a similar tendency with EG3D experiments in the paper. Without the density-blending loss, we cannot blend highly structured objects like hair. If a user does not want to reflect the reference color, we remove or give a low weight to the image-blending loss on reference: λ_2 in Eqn. 4 in the main paper.

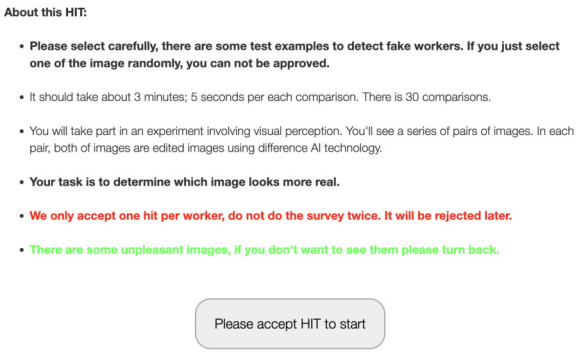

Figure 8: An introduction page of user studies in Amazon Mechanical Turk (MTurk).

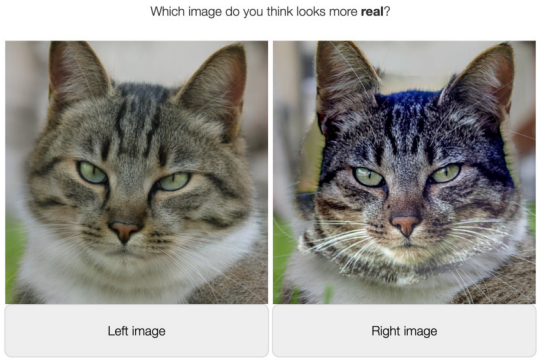

Figure 9: A comparison page between ours and baselines in MTurk.

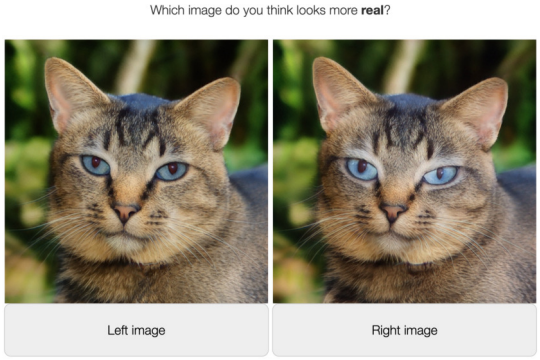

Figure 10: A comparison page of ablation study of our local alignment in MTurk.

F. User study details

We conduct extensive user studies to show the effectiveness of our methods in the realism score of human perception on the Amazon Mechanical Turk (MTurk) platform. We refer to user study pipelines of SDEdit [22] and modify the template[5] from the previous work [29]. The instruction page is shown in Figure 8, and MTurk workers participate in surveys

[5]https://github.com/phillipi/AMT_Real_vs_Fake

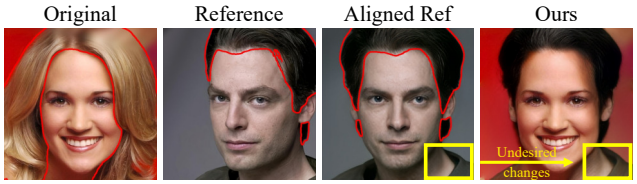

Figure 11: A failure case of blending from long hair to short hair.

comparing the results of two methods, as shown in Figures 9 and 10.

Each evaluation set consists of 25 pairwise comparison questions, with an additional five questions used to detect deceptive workers. We only invite workers with a HIT Approval Rate greater than 98%. Each set takes approximately 2 to 3 minutes and offers a reward of $0.5.

Comparison with baselines are conducted in Tables 3 and 4 of the main paper. Figure 9 shows the comparison page. There are two blending results: one for our method and the other for one of the baselines. The order of images is randomly shuffled, and a worker is instructed to select a more realistic image. In CelebA-HQ [15] and AFHQv2-Cat [9], we use 60 and 40 evaluation sets, respectively; CelebA-HQ for 1,500 and AFHQ for 1,000 pairwise comparisons. We set the same number of evaluation sets to report the combination of our method and Poisson blending.

Ablation study of 3D local alignment is conducted in Section C. The user study page is shown in Figure 10. Experimental settings are almost similar to previous studies: Tables 3 and 4 of the main paper. We show two blending results of our method with and without *local alignment*. Note that pose alignment is used in both of the results. We use 20 evaluation sets in AFHQv2-Cat; 500 pairwise comparisons.

G. Failure cases

Besides inversion as we described in the main paper, we present other failure cases in our blending method. Figure 11 shows our image blending result from a large mask to a small mask. Undesirable effects (yellow box) of the reference image have been introduced to the final result. One potential solution for future work is to inpaint the original image before blending. Figure 12 shows another failure case of local alignment in the Iterative Closest Point (ICP) algorithm [3, 8]. ICP is an approach to aligning the two point clouds: P_{ori} and P_{ref} for the original and reference objects, respectively. It iteratively minimizes the distance between each 3D point in P_{ori} and its nearest point in P_{ref}, but it may fall into the local extremum. P_{ref} shrinks to a point $\tilde{P}_{\text{ref}}$ as discussed in the previous work [27]. To mitigate this issue, we restrict the minimum and maximum scaling factor to $[0.75, 1.25]$. In the future, we may utilize recent pairwise registration techniques [10, 27] instead of ICP for better local alignment.

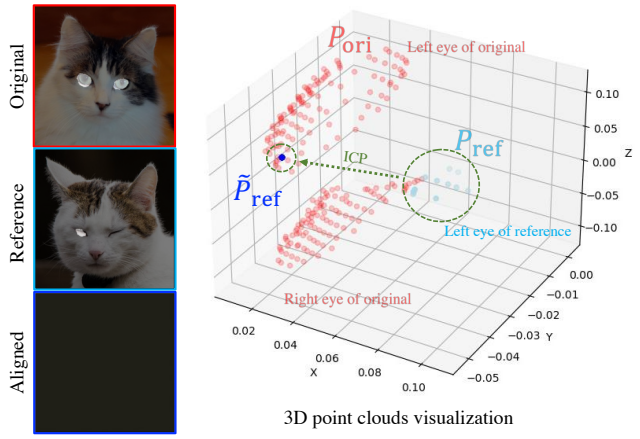

Figure 12: A failure case of the Iterative Closest Point (ICP) algorithm in Section G. If a user gives inappropriate masks or the target region of the mesh has indistinctive geometry, ICP may fail and generates a degraded aligned reference image. Masks are given as both eyes for the original image and the left eye for the reference image. The right figure shows the point cloud visualization. Red dots stand for the point cloud P_{ori} of both eyes of the original object. Light blue dots stand for the point cloud P_{ref} of the left eye of the reference object. Blue dots denote the transformed point cloud $\tilde{P}_{\text{ref}}$ of the reference after applying ICP.

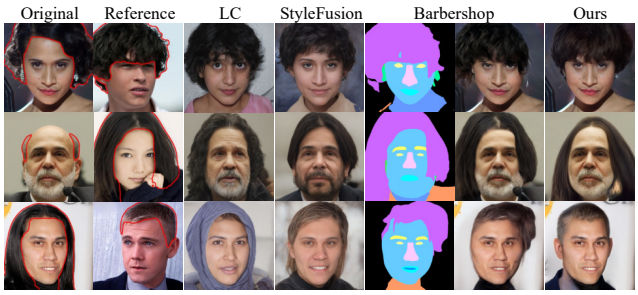

Figure 13: Additional hair blending comparison with baselines. Latent Composition (LC) generates real-looking images but far from the input images. StyleFusion alters the hair length of the reference images. In StyleFusion and Barbershop, the hair and face poses do not match in each blending result. Ours shows the best results regarding 3D-aware alignment and identity preservation.

H. Societal impact

The societal impact of image blending shares similar issues raised in the other generative models and view synthesis techniques [12]. For instance, a malicious user may use image blending to manipulate the expression and identity of a real person or create a scene that does not exist in real life. Most of the existing watermarking and visual forensics methods focus on 2D image content [28]. Adopting visual forensics methods for 3D-aware content seems to be an important future work.

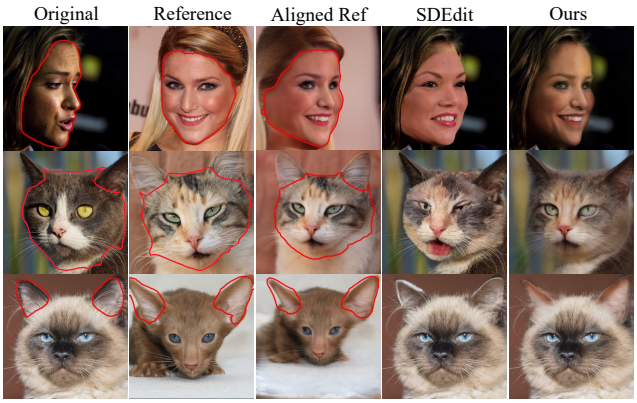

Figure 14: Comparison with SDEdit in CelebA-HQ and AFHQv2-Cat test sets. SDEdit blending results often do not match reference images well. The identities of faces or ears have been changed in both CelebA-HQ and AFHQ datasets.

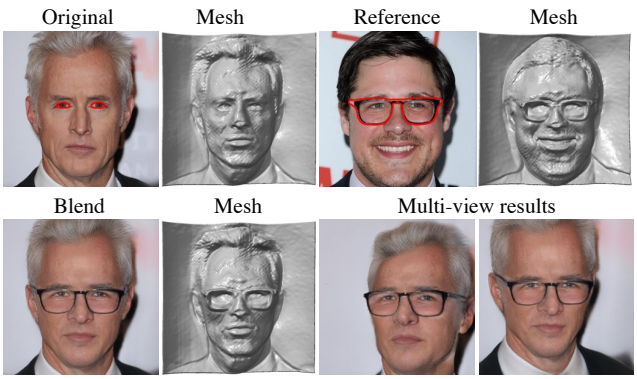

Figure 15: Blending highly structured parts such as eyeglasses. Our blending method can reflect the high-fidelity 3D shape of eyeglasses and generate multi-view consistent results.

I. Additional results

Comparison with StyleFusion and Barbershop. In order to show the advantages of our 3D-aware approach, we compare our method with additional baselines: StyleFusion [14] and Barbershop [32]. Previous methods struggle to blend fairly misaligned images, and they acknowledged this limitation in their papers. As shown in Figure 13, latent-based methods often fail to preserve identity, as projecting images into low-dimensional latent space remains challenging. For example, Latent Composition [5] and StyleFusion altered identities from the input images. Other works, such as StyleMapGAN [19] and Barbershop, use the spatial latent space to capture image details, but as a trade-off, they are worse at blending misaligned images. In contrast, our method preserves identity while maintaining 3D consistency between parts. Our method addresses these issues by 1) 3D-aware alignment and 2) blending with 3D-aware constraints, including pixel RGBs and volume density from the aligned reference.

Comparison with SDEdit in AFHQ. In addition to comparing CelebA-HQ [15], we also conduct a comparison with SDEdit using the AFHQv2-Cat dataset [9]. However, since pretrained diffusion models on AFHQ are not publicly available, we use the LSUN-Cat model[6]. Figure 14 shows SDEdit fails to preserve the reference well in both datasets.

More qualitative results. We show additional qualitative results of our 3D-aware blending. Figure 15 shows blending results of highly structured objects such as eyeglasses. Figures 16 and 17 show blending comparisons with baselines. Our results show outstanding blending results than other baselines. By virtue of NeRF, we can synthesize novel-view images. Figures 18–21 show multi-view consistent blending results. Please see our project page for the multi-view consistent blending videos.

[6]https://github.com/openai/guided-diffusion

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *ICCV*, 2019. 2, 5

[2] Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022. 1

[3] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, 1992. 2, 3, 7

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 1

[5] Lucy Chai, Jonas Wulff, and Phillip Isola. Using latent space regression to analyze and leverage compositionality in gans. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 5, 8, 10

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 5

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 12

[8] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 1992. 2, 3, 7

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 1, 3, 7, 8

[10] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *CVPR*, 2020. 7

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1

[12] Jeffrey T Hancock and Jeremy N Bailenson. The social impact of deepfakes. *Cyberpsychology, behavior, and social networking*, 2021. 7

[13] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981. 5

[14] Omer Kafri, Or Patashnik, Yuval Alaluf, and Daniel Cohen-Or. Stylefusion: A generative model for disentangling spatial segments. *arXiv preprint arXiv:2107.07437*, 2021. 8

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 7, 8

[16] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 5

[17] 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)*, 2019. 1, 5

[18] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[19] Hyunsu Kim, Yunjey Choi, Junho Kim, Sungjoo Yoo, and Youngjung Uh. Exploiting spatial dimensions of latent in gan for real-time image editing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 5, 8

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 2

[21] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Transactions on graphics (TOG)*, 1987. 2, 3

[22] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 5, 6

[23] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 5, 12

[24] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM SIGGRAPH*, 2003. 1, 5, 10

[25] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on graphics (TOG)*, 2022. 2

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016. 2

[27] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE International Conference on Computer Vision (ICCV)*, October 2019. 7

[28] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 7

[29] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 2016. 6

[30] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[31] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[32] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Barbershop: Gan-based image compositing using segmentation masks. In *ACM Transactions on graphics (TOG)*, 2021. 8

Figure 16: Blending comparison with baselines in the CelebA-HQ test set. It shows supplementary comparison results of Table 1 in the paper. LC and PB stand for Latent Composition [5] and Poisson Blending [24], respectively.

Figure 17: Blending comparison with baselines in the AFHQv2-Cat test set. It shows supplementary comparison results of Table 2 in the paper.

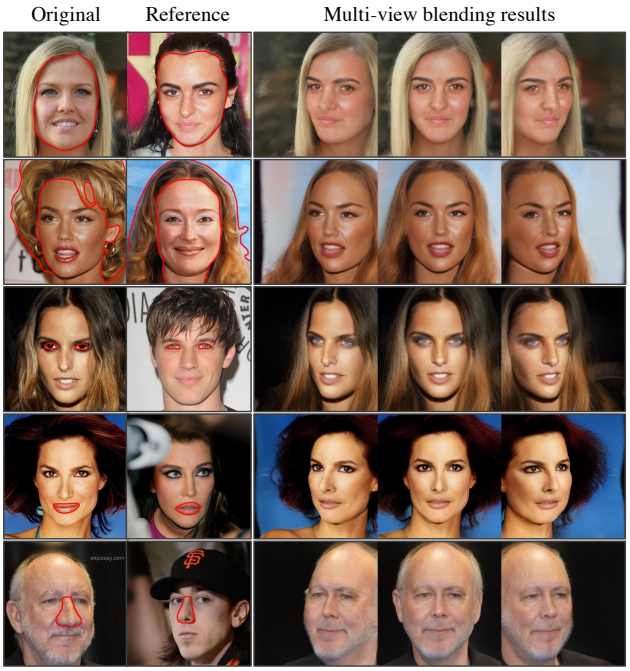

Figure 18: Multi-view blending results in CelebA-HQ using EG3D.

Figure 19: Multi-view blending results in ShapeNet-Car [7] using EG3D. The last two rows demonstrate that our method can achieve natural blending even when the sizes of the blending regions differ. Our 3D local alignment also performs well on the ShapeNet-Car dataset.

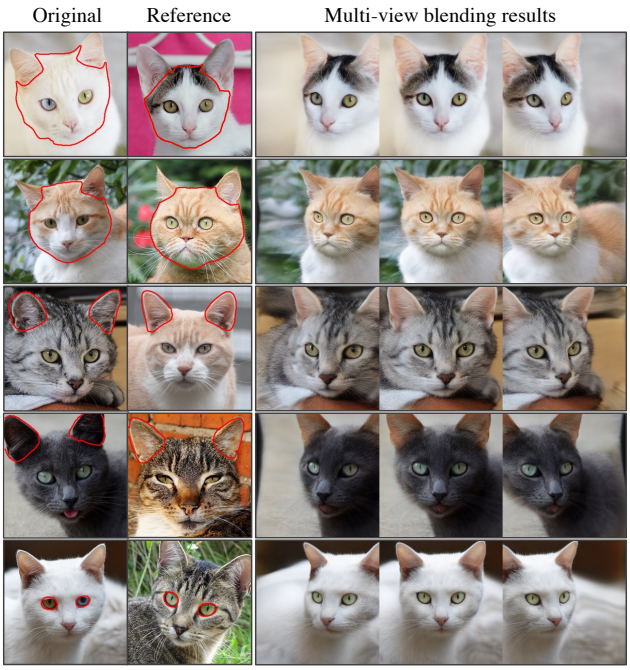

Figure 20: Multi-view blending results in AFHQv2-Cat using EG3D.

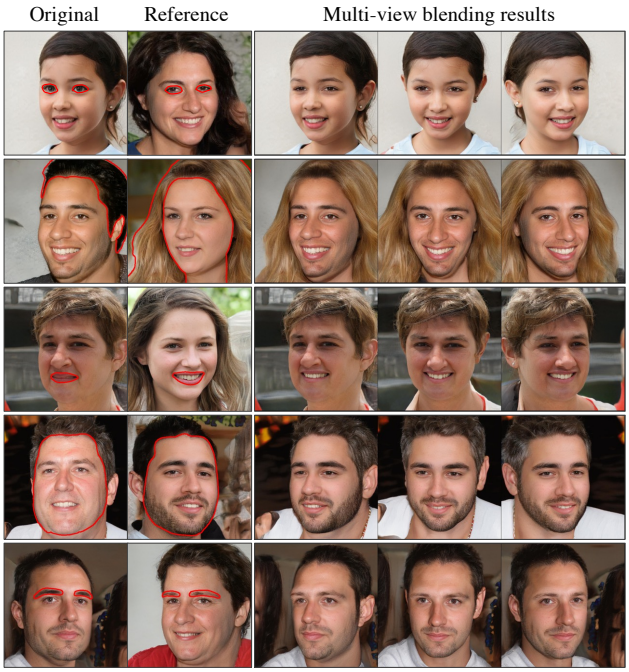

Figure 21: Multi-view blending results in generated images using FFHQ-pretrained StyleSDF [23].