Appendix

In this appendix, we provide additional explanations, experiments, and results:

- Section A: Implementation details of our guide image, mask construction, and optimization.
- Section B: Details of our FFHQ-S test set construction.
- Section C: Additional experiments.
- Section D: Additional ablation studies.
- Section E: Comparison to a concurrent work.
- Section F: Additional real-image hairstyle transfer results and examples from our user study.
- Section G: Failure cases.
- Section H: Negative societal impact

A. Implementation Details

For guide image construction and image blending, we rely on 2D facial keypoints predicted by Dlib library [22] and semantic regions predicted from a pretrained segmentation network [42, 48]. The semantic output contains 19 classes, but we group them into 6 classes: face, ear, nose, neck, hair, and background. After both input face I_f and reference hair I_h are aligned in both viewpoints, we denote by $H_{\text{hair}}, H_{\text{face}}, \ldots$ the semantic regions of I_h for the hair, face, or other parts, and analogously by $F_{\text{hair}}, F_{\text{face}}, \ldots$ for the parts in I_f.

A.1. Face-Hair Alignment

The purpose of this step is to create pose-aligned versions of I_f and I_h. That is, a new I_f in the head pose of I_h and a new I_h in the head pose of I_f. These pose-aligned I_f, I_h will be used to construct guide images. We utilize EG3D [6] for this task and use uniform scaling and translation to match their faces' widths and positions. We first explain the process to warp I_h to match the head pose of I_f. The other direction from I_f to I_h will be done similarly with a small change, discussed afterward.

A.1.1 EG3D Warping

We use EG3D [6] to rotate I_h to match I_f's pose. While EG3D projection can provide consistent geometry, the original details are often not well preserved. To fix this, we present a warping method that directly uses the original hair pixels by utilizing the EG3D estimated geometry, as shown in Figure 6. We preprocess an input image and determine the camera pose using the EG3D-proposed technique. We use the official code of EG3D with their ffhq512-64.pkl checkpoint.

Mesh Retrieval: We project I_h into EG3D's $\mathcal{W}$ latent space using PTI [29], ignoring the hat region, which can lead to inaccurate segmentation after projection. Then, we use the Marching cube algorithm [24] to construct a triangle

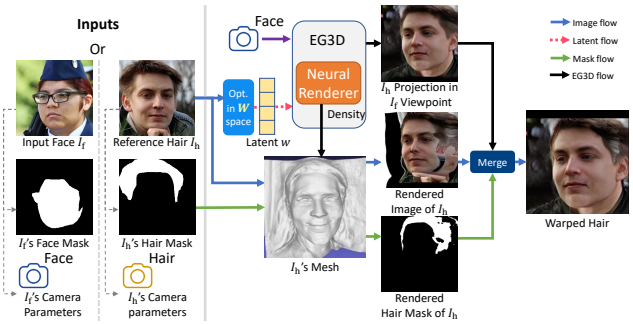

Figure 6. **Overview of EG3D warping:** We rotate I_h to the pose of I_f, and vice versa (not visualized in the diagram). This is done by combining the projection and geometry consistency of EG3D with the texture detail of the input image.

mesh from the volume density of EG3D. We assign colors to the mesh by reprojecting the mesh onto I_h and use the pixel colors from I_h. We store the triangles that belong to the hair region of I_h in a set Ψ. This will be used to determine which pixels in the mesh, after being warped into the target viewpoint, correspond to the original hair pixels from I_h.

Target-Viewpoint Rendering: We render the mesh in the 512x512 resolution in I_f's pose and replace all pixels outside the hair region of I_h (not in Ψ) with the I_h's projection that is warped to the I_f's pose by EG3D. We repeat the same process to warp I_f to I_h, but switching their roles and change the set Ψ to contain the face region of I_f instead.

A.1.2 Uniform Scaling and Translation:

We align I_h with I_f by uniform scaling and translation to match the faces' widths and centers. The face width is calculated as the difference in the x-coordinate between the right-most (k_{16}) and left-most (k_0) keypoints. The x and y center coordinates are computed separately. The x-center is the x coordinate of $(k_0 + k_{16})/2$ and the y-center is the y coordinate of $((k_0 + k_{16})/2 + k_8)/2$. This improves alignment of faces with larger pose differences.

A.2. Color Fill-In in the Guide Image Construction

To construct I_{guide}, there are some corner cases that need to be properly handled, such as when the existing hair in the input is larger than the reference hair. Fortunately, addressing most corner cases amounts to handling the following four scenarios, as shown in Figure 7.

1. If the existing hair shape in I_f is larger than the reference hair in I_h, fill the hair region not overlapped by the reference hair with the average background color.

2. If I_f has bangs on the forehead, remove them by filling that region with the average skin color.

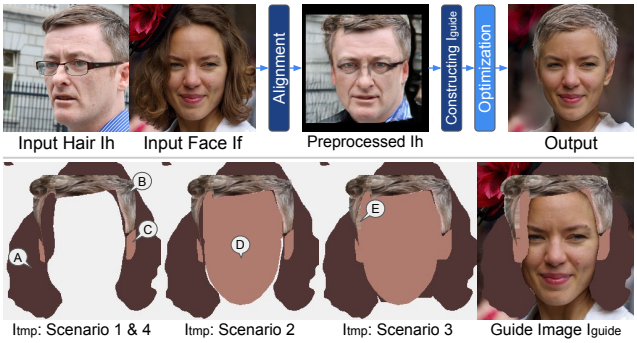

Figure 7. **Guide image construction.** We create a guide images I_{guide} with a simple cut-and-paste of the hair from I_h onto I_f with a few modifications to handle the four scenarios in Section A.2. (We also create another guide image by transferring the face from I_f to I_h, but not visualized by this diagram.) We solve this by first creating a temporary canvas image I_{tmp} to remove the original hair in I_f (Scenario 1; A,B), transfer the visible ears from I_h to I_{guide} (Scenario 4; C), and remove I_f's bangs (Scenario 2; D) while preserving the face width of I_f (Scenario 3; E). Then, we combine I_{tmp} with I_f and I_h to create I_{guide}.

3. If I_f has a narrower face than I_h, the guide image should retain the face width of I_f and fill in both sides next to the face with the hair color of I_h.

4. If the ears are visible in I_h, transfer the ear regions to the guide image and fill those regions with the skin color of I_f.

We first create a temporary canvas image I_{tmp} as follows. We fill the region F_{hair} in I_{tmp} with the average color of the pixels within F_{bg} (background) in I_f (Scenario 1), and copy H_{hair} in I_h to I_{tmp}. Then, if H_{ear} exists (Scenario 4), fill the region H_{ear} in I_{tmp} with the skin color of I_f. We approximate the skin color by averaging the nose pixels of the I_f in F_{nose}. Then, to remove the existing hair (Scenario 2), we fill the face area of I_{tmp} defined by the area above keypoints $k_0 - k_{16}$ with the I_f's skin color. This face area is denoted by $F^{\text{k}}_{\text{face}}$. We then fill I_{tmp} in the region $F^{\text{k}}_{\text{face}} \cap (H_{\text{face}} \cup H_{\text{neck}})$ with the I_f's skin color. In the case where I_f has a narrower face (Scenario 3), we fill $(H_{\text{face}} \cup H_{\text{neck}}) - F^{\text{k}}_{\text{face}}$ with the hair color of I_h in a row-by-row basis. After I_{tmp} has been created, our I_{guide} is constructed by first setting $I_{\text{guide}} = I_f$, then copying the content in the region F_{hair} of I_{tmp} to I_{guide}. Lastly, when some part of H_{hair} overlaps with F_{face}, it is unclear whether the overlapped region should be hair or face in I_{guide}. To solve this, we update H_{hair} by removing any region of H_{hair} that lies outside of the face region defined by the detected keypoints. With this updated H_{hair}, we copy the hair of I_h in this region to I_{guide} to finish its construction.

A.3. Mask Construction

We show the masks used in each loss function in Figure 8.

$\boldsymbol{M}^{\text{f}}_{\text{roi}}$; $\boldsymbol{M}_{\text{f}}$ represents the face region of I_{guide}, computed by $F_{\text{face}} - H_{\text{hair}} - H_{\text{hat}}$. Additionally, we erode the region in M_{f} that is higher than the eyebrows (5 pixels above the highest keypoints) using 5 iterations.

$\boldsymbol{M}^{\text{h}}_{\text{roi}}$; $\boldsymbol{M}_{\text{h}}$ represents the hair region of I_{guide}, computed by $\textit{erode}(H_{\text{hair}}, 5)$. (I.e., eroding H_{hair} using 5 iterations)

$\boldsymbol{M}^{\text{bg}}_{\text{roi}}$; $\boldsymbol{M}_{\text{bg}}$ represents the background region in I_f that is not covered by the transferred hair, computed by $F_{\text{bg}} - \textit{dilate}(H_{\text{hair}} \cup M_{\text{out}}, 5)$, where M_{out} represents out-of-the-frame regions. (Suppose, for example, H_{hair} extends down to the bottom edge of I_h, the entire region below it in M_{out} will be marked 1). Because there is no accurate background in the I_h viewpoint, every pixel in $M^{\text{hair}}_{\text{bg}}$ is zero.

$\boldsymbol{M}^{\text{f}}_{\text{rnoi}}$ represents the face region in I_f that was previously occluded but should be visible in the final output, computed by $F^{\text{k}}_{\text{face}} + H_{\text{ear}} - M_{\text{f}}$.

$\boldsymbol{M}^{\text{h}}_{\text{rnoi}}$ represents the hair region that was previously occluded by other objects or not visible due to image cropping, computed by $H_{\text{hat}} \cup H_{\text{face}} \cup H_{\text{neck}} \cup M_{\text{out}}$.

$\boldsymbol{M}_{\text{c}}$ represents the regions in I_{guide} that were copy-pasted from I_f or I_h (including warped pixels), computed by $\textit{erode}(M_{\text{f}} \cup M_{\text{h}} \cup (F_{\text{bg}} \cap O^1_{\text{bg}}), 5)$, where O^1_{bg} represents the background region in O_1.

$\boldsymbol{M}_{\text{raw}}$ represents the regions in I_{guide} that were copy-pasted from I_f or I_h with the original pixel content. The mask $M^{\text{face}}_{\text{raw}}$ is computed by $\textit{erode}(M_{\text{bg}} \cup M_{\text{f}}, 10)$, and $M^{\text{hair}}_{\text{raw}}$ is $\textit{erode}(M_{\text{h}}, 5)$.

A.4. Regularization Losses

This section elaborates on the regularization losses described in prior work.

Noise Regularization Loss: This loss proposed in StyleGAN2 [19] is used ensure that the noise maps capture only the stochastic variations, by encouraging the optimized noise maps to be normally distributed via minimizing the normalized spatial autocorrelation:

$$\mathcal{L}_{\varepsilon} = \sum_{i,j} \Big(\text{mean}(n^j_i \odot H(n^j_i))^2 + \text{mean}(n^j_i \odot V(n^j_i))^2 \Big), \tag{11}$$

where $H(\cdot)$, $V(\cdot)$ shift the noise map horizontally/vertically by one pixel with wrap-around edges. And for each noise map n_i, the autocorrelation is computed for different downscaled versions $n^0_i, n^1_i, \ldots$ down to the 8x8 resolution.

PTI Regularization Loss: This loss proposed in PTI [29] restricts any change in the latent space to a small area. In each iteration, we sample w_z from $\mathcal{W}$ space, then create an interpolated latent code w_r between w_z and the optimized latent code w with an α parameter.

$$w_r = w_{\text{optimized}} + \alpha \frac{w_z - w_{\text{optimized}}}{\|w_z - w_{\text{optimized}}\|_2} \tag{12}$$

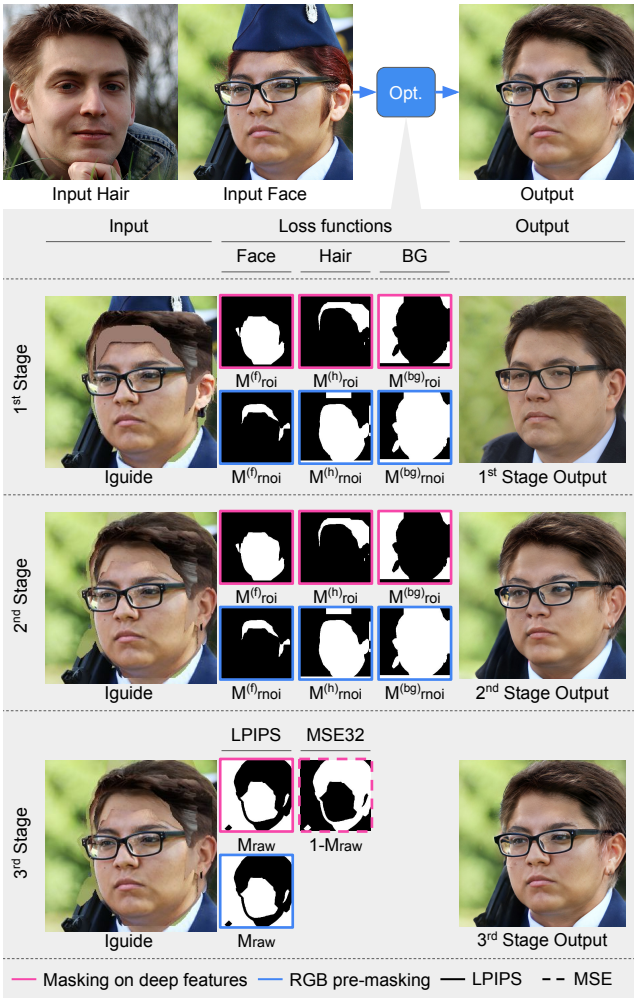

Figure 8. Masks used in each loss function. Masks used on the deep features in LPIPS are in pink frames, and masks used for RGB-premasking are in blue frames.

To compute the final loss value, we feed the latent code w_r into the original StyleGAN to produce R and into a weight-tuned StyleGAN to produce R^*.

$$\mathcal{L}_{\mathrm{r}} = L_{\mathrm{LPIPS}}(R, R^*) + L_{\mathrm{MSE}}(R, R^*) \tag{13}$$

A.5. Optimization Details

For our $\mathcal{W}$ and $\mathcal{W}^+$ latent optimization, we use Adam optimizer [23] with $(\beta_1, \beta_2) = (0.9, 0.999)$. We run the optimization for 1,000 iterations in the first stage, and 500 iterations in the second stage with the same learning schedule used in StyleGAN2 [19]. The learning rate is ramped up linearly from 0 to 0.1 during the first 5 percent of iterations (50/25) and ramped down with a cosine schedule during the last 25 percent of iterations (250/125). The initial w_0 used in the first stage is computed by averaging 10,000 latent codes (Section 3.2.1).

The parameter $\Lambda^{(\mathrm{i})}$, which is used to scale $\mathcal{L}_{\mathrm{per}}^{[\mathrm{f/bg}]}$ in I_f viewpoint and $\mathcal{L}_{\mathrm{per}}^{\mathrm{h}}$ in I_h viewpoint, is set to 6 in the first stage, 4 in the second stage. (The other losses are scaled by 1). The parameters $(\lambda_{\mathrm{p}}^{\mathrm{f}}, \lambda_{\mathrm{p}}^{\mathrm{h}}, \lambda_{\mathrm{p}}^{\mathrm{bg}}, \lambda_{\mathrm{g}}, \lambda_{\mathrm{i}}, \lambda_{\varepsilon}, \lambda_s)$ are set to $(2, 1, 0.66, 2, 4, 10^5, 3)$ in the first stage, $(1, 2, 1, 2, 4, 10^5, 2)$ in the second stage.

We perform PTI [29] optimization in the third stage for 500 iterations. We multiply L_{LPIPS} with 2 and use the default PTI parameters.

For PTI in the EG3D projection, we follow EG3D's optimization procedure, which runs 500 iterations for $\mathcal{W}$ latent optimization and 500 iterations for tuning.

A.6. Running Time Comparison

We measured our runtime on a single GPU NVIDIA RTX 2080Ti with AMD Threadripper 2920x. We used around 21 minutes per input pair. The construction time for the guide images is around 13 minutes: 5 minutes for EG3D projection and 6–10 minutes for EG3D warping. Multi-view latent optimization requires around 8 minutes.

Note that we have not optimized our code, and many of the 3D pre-processing steps (8 mins), such as our occlusion test and marching cube, can be implemented on the GPU with real-time speed (currently, it's in python). Techniques such as PSP [28] can speed up and perform our EG3D projection with a single network inference. The rest of the pipeline takes about 8 mins, which is in the same order as LOHO: 15 mins, Barbershop: 5 mins, and StyleYourHair: 8 mins. HairNet still does require StyleGAN projections and PTI [29] for pre-precessing, which take several minutes per image.

B. Construction of Our FFHQ-S Testset

This section explains the criteria used for determining the four scenarios in 12-Config FFHQ-S in Table 1. All criteria are computed from raw I_f and I_h. We skip any pair in which the number of hair pixels in I_h is less than 5 percent of all the pixels in the image.

Pose Misalignment: The criterion for this scenario is based on the difference between the yaw angles of I_f and I_h, estimated from facial keypoints [16]. The angle difference between [0,15) is indicated with '-', [15, 30) with a checkmark, and [30, 45) with double checkmarks in Table 1.

Needs Face Inpainting: This criterion tests whether I_f's face is occluded, by checking if the number of pixels in $H_{\mathrm{face}} - F_{\mathrm{face}}^{\mathrm{k}}$ is greater than 10 percent of all pixels.

Needs BG Inpainting: This criterion tests whether a substantial number of background pixels need to be hallucinated. This happens when I_f's hair is smaller than I_h's hair or, specifically, when the number of pixels of H_{hair} - F_{hair} is greater than 15 percent of all pixels.

***I_h* Contains Hat:** This criterion tests whether some part of the reference hair in I_h is missing due to hat wearing by

Figure 9. Results when StyleYourHair [20] and Barbershop [46] use our rules to create the target segmentation mask: Bangs and unnecessary hair are totally removed (A). However, the modified target segmentation is not realistic (similar to our guide) and produces unrealistic hair shapes (B) with poorer color reproduction (C). Although our modified target segmentation can improve Barbershop's result in the second row, it cannot improve StyleYourHair, whose technique also includes a hair-warping stage (D).

checking if the number of pixels in H_{hat} is greater than 5 percent of all pixels.

C. Additional Experiments

C.1. Can Prior Work Solve Challenging Scenarios With a Good Target Segmentation Mask Constructed Using Our Rules?

In this section, we construct a target segmentation mask based on our rules in Section 3.1 / Appendix A.2 and use it in place of the original mask used in StyleYourHair [20] or Barbershop [46], then compare their results with ours.

Figure 9(A) shows that the original Barbershop and StyleYourHair fail to completely remove bangs from the forehead or add more hair that makes the hairstyle incorrect, but the modified version using our provided target segmentation mask can remove the bangs completely as well as any unnecessary hair. Compared to our method, this modified version still produces (B) unnatural hairstyles with (C) poorer color reproduction.

Unlike Barbershop and StyleYourHair, which use a high-dimensional latent space that can overfit the error-prone target segmentation mask, our method can better refine the boundaries between semantic regions by first predicting the output in original latent space that ensures natural-looking hair before refining the output in the extended space with LPIPS pre-masking for seamless blending. Importantly, this shows that our state-of-the-art quality requires not only our well-designed guide image but also our multi-view latent optimization that uses the guide image in a flexible and effective manner.

C.2. Quality of Hairstyle Transfer

Following StyleYourHair [20], we compute the FID score to compare the distributions of the results and real images.

	LOHO	Barbershop	StyleYourHair	Ours
Hairstyle (FID ↓)	**20.72**	21.22	21.64	<u>21.02</u>

Table 4. FID scores after performing the hairstyle transfer (Section C.2).

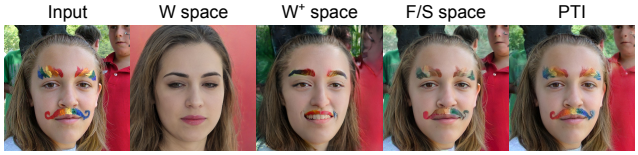

Figure 10. Comparison of different reconstruction techniques ordered by the degree of freedom. PTI with the highest degree of freedom successfully reconstructs the texture and color details of a woman with heavy makeup, as shown in this example.

However, note that FID is not ideal for this task because an algorithm that minimally changes or does not change any hair at all can achieve the best performance. We use the same dataset in Section 4.4. All methods yield comparable results and are roughly equivalent, shown in Table 4. LOHO, which has face and background blending, receives the best FID score of 20.7. However, according to our user study, people are less likely to prefer LOHO over other methods.

C.3. Comparison of the ability to preserve details.

To evaluate the effectiveness of each method in preserving image details, we perform an image reconstruction task without employing the hairstyle transfer technique, in order to eliminate any potential external factors. Figure 10 demonstrates that the ability to preserve details highly depends on the degree of freedom (sorted descendingly): pivot tuning inversion (PTI), $\mathcal{F}/\mathcal{S}$ space, $\mathcal{W}^+$ space, and $\mathcal{W}$ space.

D. Additional Ablation Studies

D.1. Qualitative Ablation Studies

Ablation studies with quantitative metrics are highly difficult to do because there is no ground truth and the existing metrics such as FID score are unreliable. For example, dropping the latent sharing (Figure 4a) can produce a better FID score despite the clearly wrong hairstyle, which is not captured by FID. Nonetheless, we identified three most crucial components and conducted an additional user study (30 randomly sampled input pairs, each evaluated by 3 different users). The users preferred our full method 27.9% of the time, compared to Config i) 23.4%, ii) 4.5%, iii) 25.2% in Figure 4.

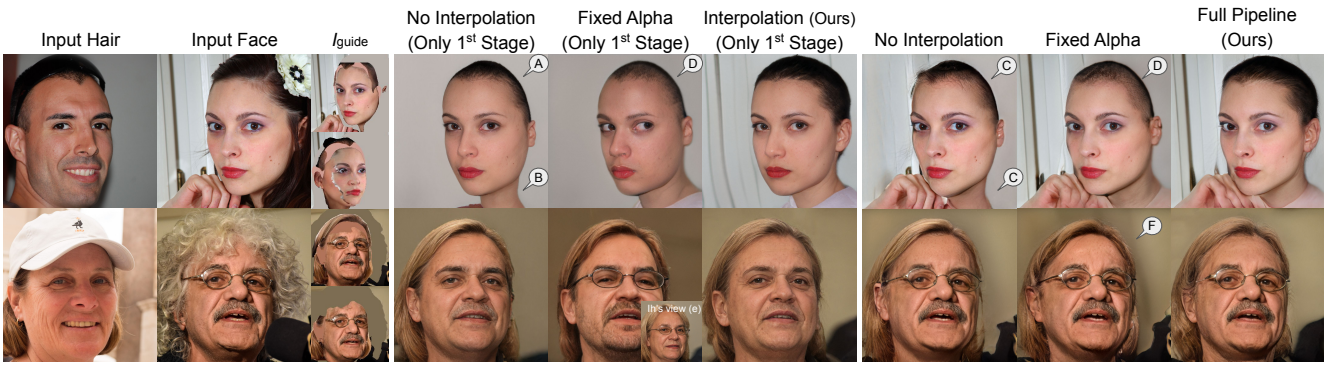

Figure 11. Ablation study on $\mathcal{W}$ space latent optimization. Without our latent interpolation technique, the results are less realistic (A,B,C,D). If we fix the interpolation coefficient α, some information, such as color, may not be shared between two viewpoints (E), resulting in poorer background details (F).

D.2. Ablation studies on latent sharing and the loss function.

We test additional ablation configurations on our multi-view $\mathcal{W}$ space latent optimization: i) sharing w latent code without interpolation, and ii) fixing α to 0.5. The results are shown in Figure 11. We also test our complete pipeline iii) without using L^{32}_{MSE} and show the results in Figure 12.

Instead of sharing the latent code with our interpolation technique, we optimize a shared latent code, which is fed to the last l layers of StyleGAN ($w_{l:18}$) for optimizing both views (Config i). This is similar to the concept of sharing w^+ latent code in Section 3.2.2, but with fewer parameters (19x512 fewer than w^+ code, and 512 more than w code). We add the latent similarity loss (see Section 3.2.1) between the latent codes used for StyleGAN's early layers and the new latent code to ensure that all latent codes are similar, which can avoid overfitting.

In Figure 11, Config i) may produce unrealistic head shapes (Figure 11-A) or necks (Figure 11-B in the first stage. This artifact still manifests in the second and third stages of optimization (Figure 11-C. This Config i) has a higher degrees of freedom and thus can fit unrealistic guide images.

When we fix the interpolation coefficient α (Config ii), the results also contain unrealistic head shapes (Figure 11-D), and the colors of the face, hair, or background may look different in each viewpoint (Figure 11-E). As a result, the hallucinated background from this configuration becomes less accurate (Figure 11-F). Our proposed method helps alleviate this issue by forcing the shared part to be similar via random interpolation. In particular, the optimizer is encouraged to use the same values for both latent codes so that their interpolation with any α will remain stationary.

Without L^{32}_{MSE} (Config iii), the optimization in each stage would not try to reproduce the overall appearance of I_{guide} and is free to synthesize arbitrary content on regions not constrained by any loss function. This can result in more realistic background details (Figure 12-A) but less realistic shading (Figure 12-B) or excessive hair (Figure 12-C).

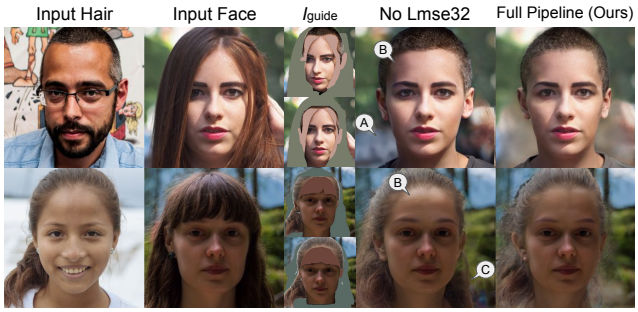

Figure 12. Ablation study on L^{32}_{MSE}. Without this loss, the regions not constrained by any loss function can be hallucinated freely. While this can lead to some positive results, such as better background details (A), it can also produce unrealistic shading (B) or excessive hair (C).

E. Comparison to concurrent work, HairNet

HairNet is also capable of pose-invariant hairstyle transfer. Unfortunately, their official code is not publicly available during our study. We provide a qualitative comparison in Figure 13 and conducted a user study on their selected input pairs (380 input pairs, each evaluated by 3 different users). The participants preferred our results 52.4% of the time, whereas HairNet was selected for 47.6%.

We observe that HairNet often fails to preserve input face identity or hair details (Figure 13, top row). Our method excels at rotating the input hair (left side, 2nd row) and restoring unseen facial features (2nd row, right side). Conversely, HairNet is better at filling in background details (left side, last row) and producing realistic hair and lighting details (right side, last row).

In addition, we calculate the maximum, minimum, and average pose difference of HairNet's input pairs, which are 22.8, 0.0, and 4.7, respectively. We suggest using our datasets for further analysis of the results in future work.

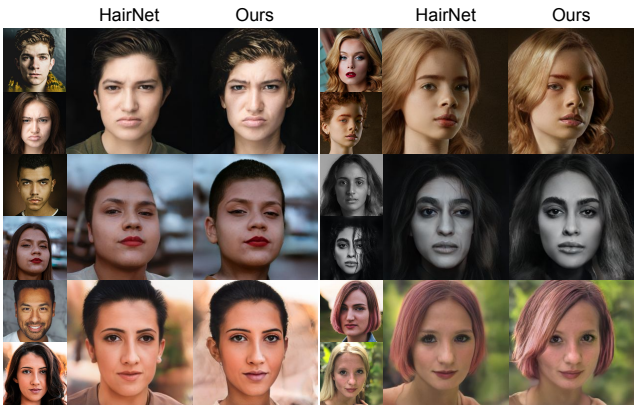

Figure 13. Comparison to concurrent work, HairNet. Our method is better at preserving input face identity and input hair details (top row), hair rotation (left side, 2nd row), and face inpainting (right side, 2nd row). Conversely, HairNet is better at background inpainting (left side, last row) and producing realistic hair and lighting details (right side, last row).

F. Additional Results

In this section, we present more qualitative results in Figure 14, 15, and random test samples from our user study in Figure 16, 17, and 18.

G. Failure Cases

Figure 5 compiles a set of our failure cases. We cannot transfer hairstyles with incorrect semantic regions (Figure 5-A). Large errors from the keypoint detector can place the hair in the wrong place (Figure 5-B). Poor EG3D projection results may make the hair look different from the reference hair (Figure 5-C). Some hair may be blended into the background if the reference hair has a similar color as the background (Figure 5-D). Other failure cases include mismatched lighting conditions (Figure 5-E) and highly unusual hairstyles (Figure 5-F).

H. Potential Negative Societal Impact

Even though the hairstyle transfer results from our method are realistic, they are considered *fake* images and can have similar uses and misuses as DeepFake. The results may contain some artifacts that another network can easily detect [10]. Our method relies on multiple pretrained networks, which may contain race, gender biases. Our method also may not work as well on people who are less represented in the training set.

We also consider the validity of the copyright for the reference hair after it has been transferred to our input, as well as the possibility of transferring the hairstyles of others without their permission. To circumvent this, we suggest that the use of both input face and reference hair be under Creative Commons licenses to prevent any potential conflicts.

Figure 14. Our method can transfer the hairstyle from any reference hair image in the top row to an input person [4, 9, 27, 31, 34, 36, 38, 39] in each row.

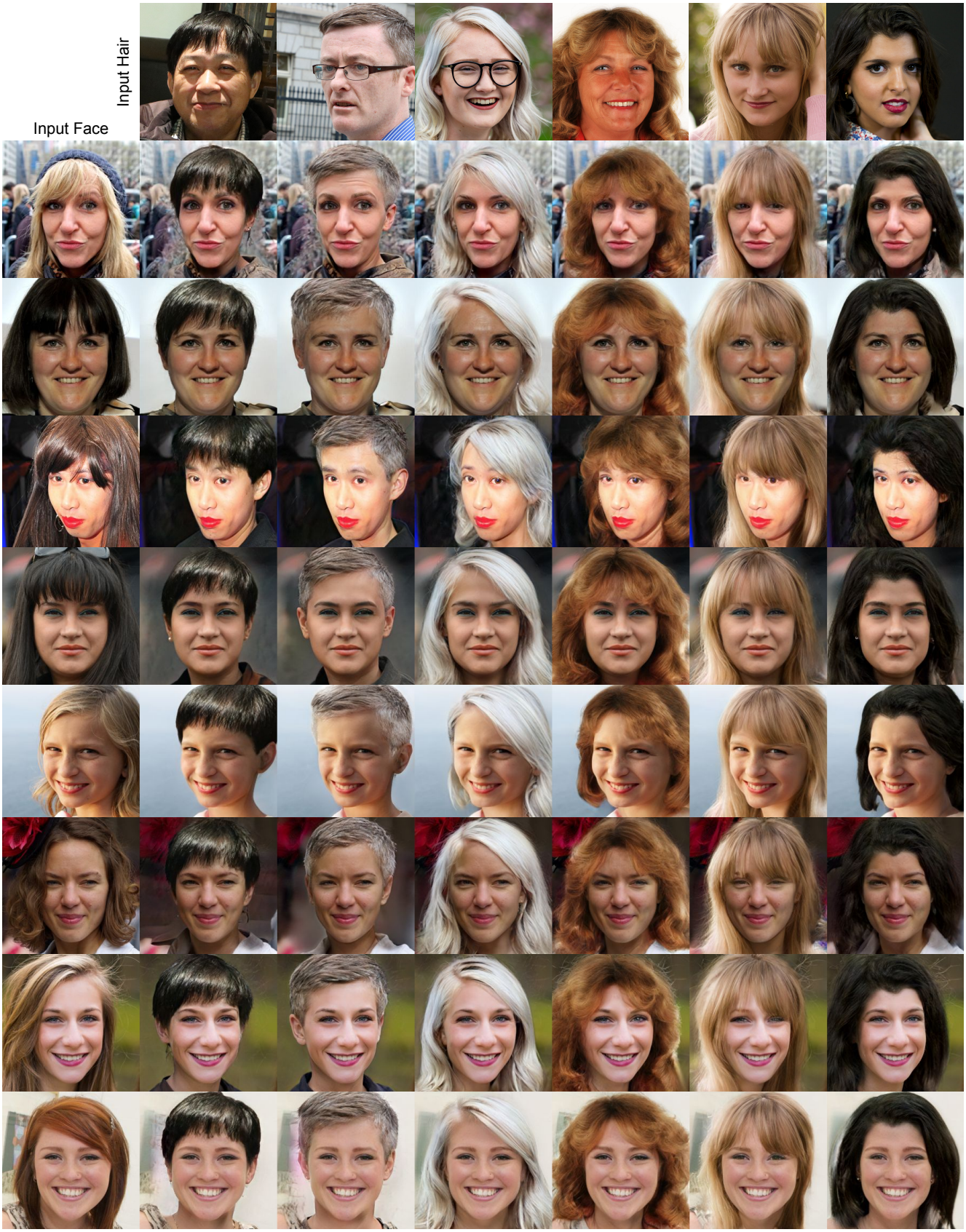

Figure 15. Our method can transfer the hairstyle from any reference hair image in the top row to an input person in each row.

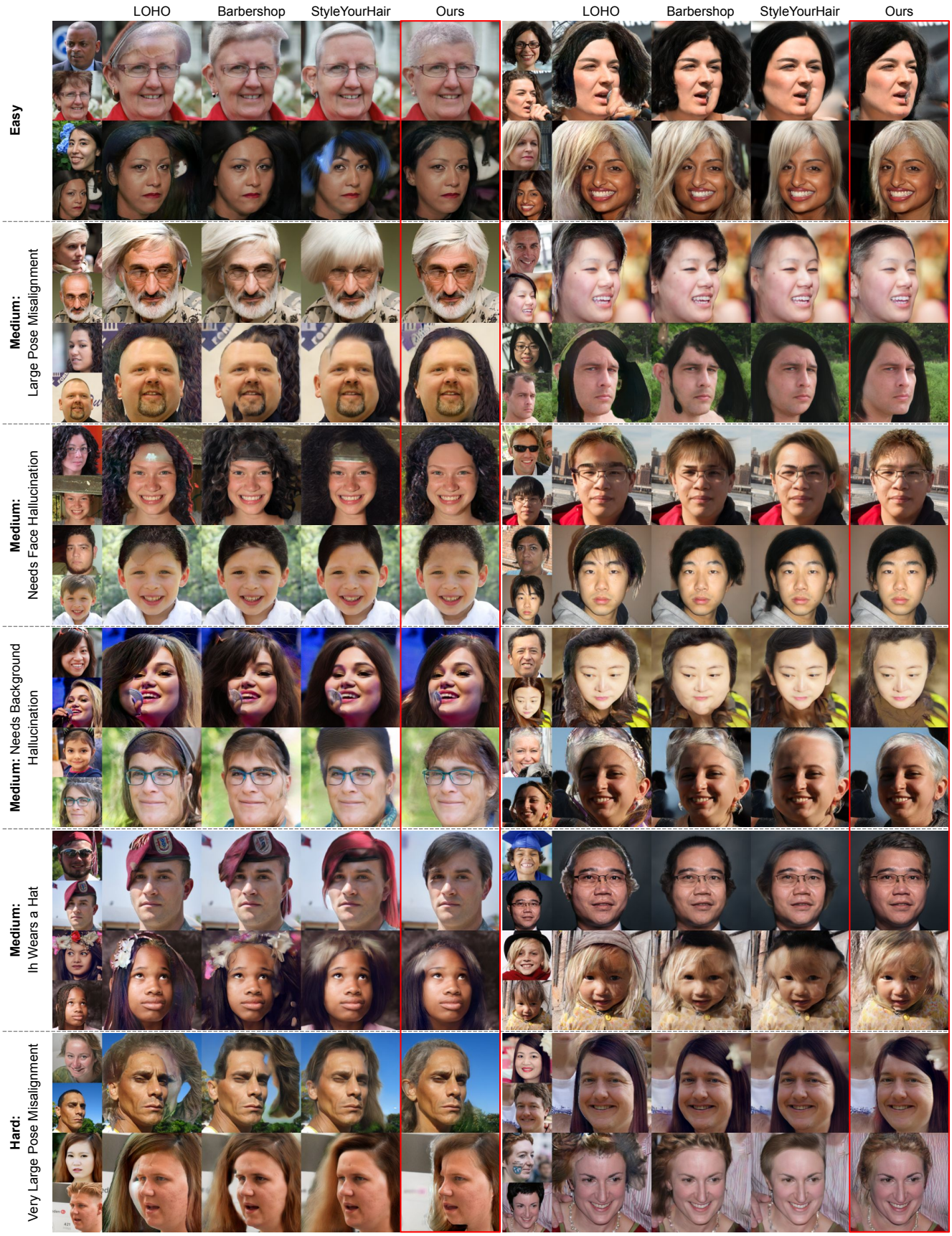

Figure 16. Random test samples from FFHQ-S for comparison to StyleYourHair [20], Barbershop [46], and LOHO [30].

Figure 17. Random test samples from FFHQ-S for comparison to StyleYourHair [20], Barbershop [46], and LOHO [30].

Figure 18. Random test samples from FFHQ-P for comparison to StyleYourHair [20], Barbershop [46], and LOHO [30].