Exploiting Spatial Dimensions of Latent in GAN for Real-time Image Editing -Supplementary Material-

Hyunsu Kim[1][2] Yunjey Choi[1] Junho Kim[1] Sungjoo Yoo[2] Youngjung Uh[3]

[1]NAVER AI Lab [2]Seoul National University [3]Yonsei University

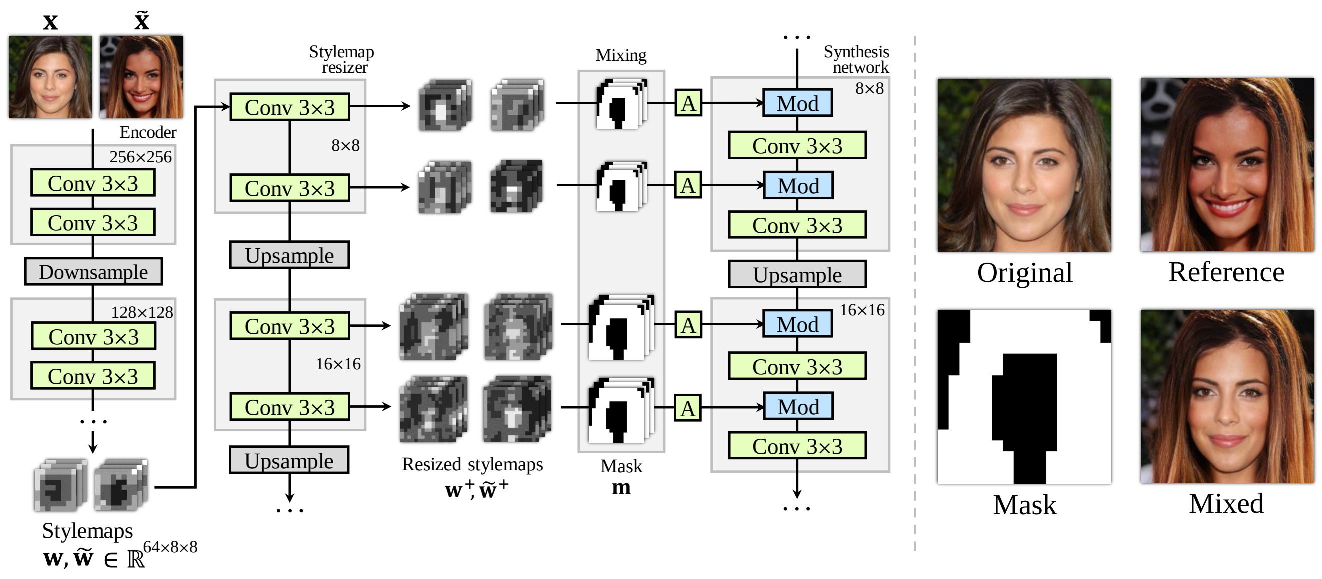


Figure 1: Our local editing starts with a learned encoder for fast image-to-stylemap projection. We estimate the stylemaps $\mathbf{w}$ and $\widetilde{\mathbf{w}}$ of the original $\mathbf{x}$ and the reference $\widetilde{\mathbf{x}}$ and transform them into multiple resolutions through the learned stylemap resizer. For each resolution, we calculate the alpha blending of the two stylemaps using the user-defined binary mask $\mathbf{m}$. Finally, the learned generator produces the output using the spatially-mixed stylemaps. The right one shows an example generated using our method.

A. Local editing in $\mathbf{w}^+$ space

This section illustrates how we perform local editing using StyleMapGAN. Although we already described the local editing method in Section 3.3 of the paper, it is impossible to edit in detail due to the coarse mask resolution (8×8). Contrary to the previous method, we propose a local editing method in $\mathbf{w}^+$ space. Regardless of the resolution of stylemap ($\mathbf{w}$), we can exploit detailed masks with resized stylemaps ($\mathbf{w}^+$) in high resolutions.

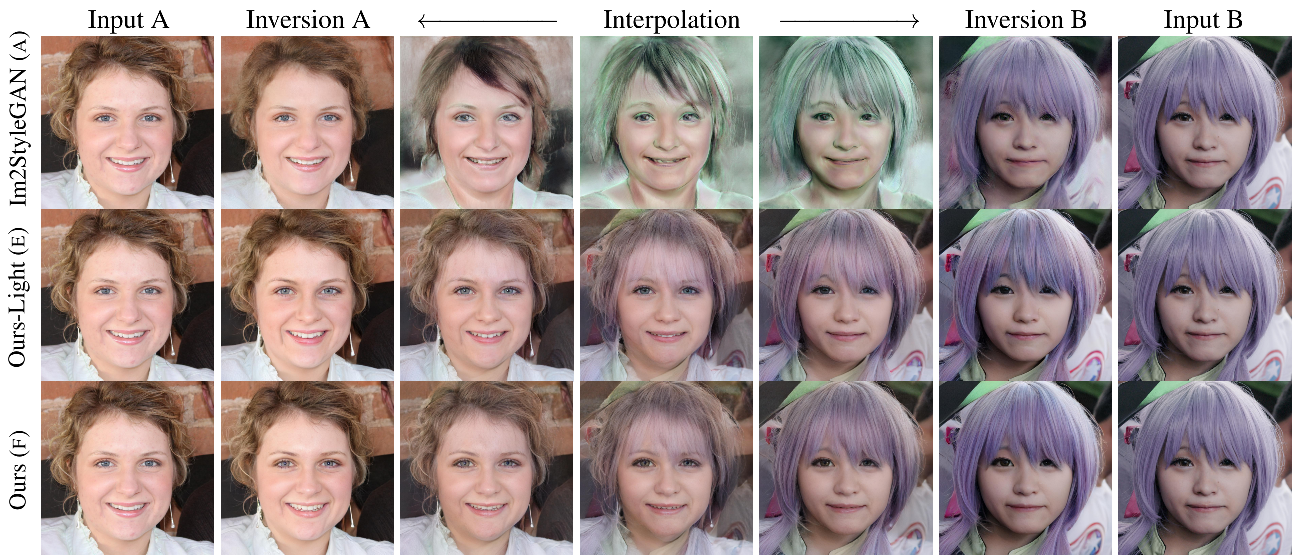
Figure 1 shows the overview of blending on the $\mathbf{w}^+$ space. The edited i-th resized stylemap $\ddot{\mathbf{w}}^+$ is an alpha blending of $\mathbf{w}^+$ and $\widetilde{\mathbf{w}}^+$:

$$\ddot{\mathbf{w}}_i^+ = \mathbf{m}_i \otimes \widetilde{\mathbf{w}}_i^+ \oplus (1 - \mathbf{m}_i) \otimes \mathbf{w}_i{}^+ \qquad (1)$$

where i-th resized mask $\mathbf{m}_i$ is shrunk by max pooling. Although the mask's shape does not align with the 8×8 stylemap, we can precisely blend the two images on the $\mathbf{w}^+$ space.

B. Experiments in the high-resolution dataset

We evaluate our model on FFHQ at 1024×1024 resolution. Baseline is StyleGAN2, and we also test Image2StyleGAN (A). StyleGAN2 official pretrained network is used for a fair comparison. StyleMapGAN adopts 32×32 stylemap for the high-resolution dataset, compared to 8×8 stylemap for 256×256 image. StyleMapGAN-Light (E) is a light version of StyleMapGAN; it reduces the number of parameters of the generator. Another training setting (D) is sequential learning, which trains the generator first and then trains the encoder. In Table 1, we used the same protocol as



Network	Projection	Joint learning	G param(M)	runtime(s)	G GPU(GB)	MSE	LPIPS	FID_{lerp}
A StyleGAN2	Image2StyleGAN	✗	30.4	454.7	2.1	0.021	0.468	38.00
B StyleGAN2	StyleGAN2	✗	30.4	142.2	2.1	0.093	0.467	34.65
D StyleMapGAN-Light	Encoder	✗	18.6	0.253	3.0	0.071	0.546	201.18
E StyleMapGAN-Light	Encoder	✓	18.6	0.253	3.0	0.017	0.347	**13.52**
F StyleMapGAN	Encoder	✓	46.4	0.249	3.1	**0.016**	**0.344**	13.68

Table 1: Comparison with StyleGAN2 on 1024×1024 FFHQ. We also explored the effect of other components such as generator size and joint learning. StyleMapGAN-Light has reduced the number of channels of the stylemap resizer. 32×32 stylemap is used for the large size of the image. “G” denotes the generator.

the paper to calculate MSE, LPIPS, and FID_{lerp}. The number of training images for FFHQ is 69K, and we limited the test and validation set to 500 images.

Comparison with baselines. As shown in Table 1, Image2StyleGAN reconstructs the image well, but it struggles with poor interpolation quality. Low FID_{lerp}, rugged interpolation results and lengthy runtime shows Image2StyleGAN is not suitable for image editing tasks. StyleMapGAN outperforms baselines in all metrics, and even StyleMapGAN-Light shows astonishing results.

StyleMapGAN-Light is 2.5× smaller than the original version. Stylemap resizer accounts for a large portion of the network’s size, so we reduce the number of channels of feature maps in the stylemap resizer. The reconstruction image lacks some detail, but StyleMapGAN-Light still outperforms baselines, and FID_{lerp} is even better than the original version. Please see our code to refer to the number of channels.

Joint learning is important when training StyleMapGAN. It makes training stable and network performance better. Training the encoder after training the generator fails to reconstruct images. We speculate the reason why joint learning is better than sequential learning as follows. In joint learning, the generator and the encoder affect each other. The generator generates an image that would be easy to reconstruct by the encoder. The structure of the encoder is a stack of convolutional layers, which makes the projected stylemap is prone to have local correspondence: Partial change in the stylemap leads to local editing on the image. Through joint learning, the mapping network in the generator also learns to make the stylemap from Gaussian distribution have the local correspondence.

C. Implementation details

Architecture. We follow StyleGAN2 [12] regarding the discriminator architecture and the feature map counts in the convolutional layers of the synthesis network. Our mapping network is an MLP with eight fully connected layers followed by a reshape layer. The channel sizes are 64, except the last being 4,096. Our encoder adopts the discriminator architecture until the 8×8 layer and without minibatch discrimination [20].

Training. We jointly train the generator, the encoder, and the discriminator. It is simpler and leads to more stable

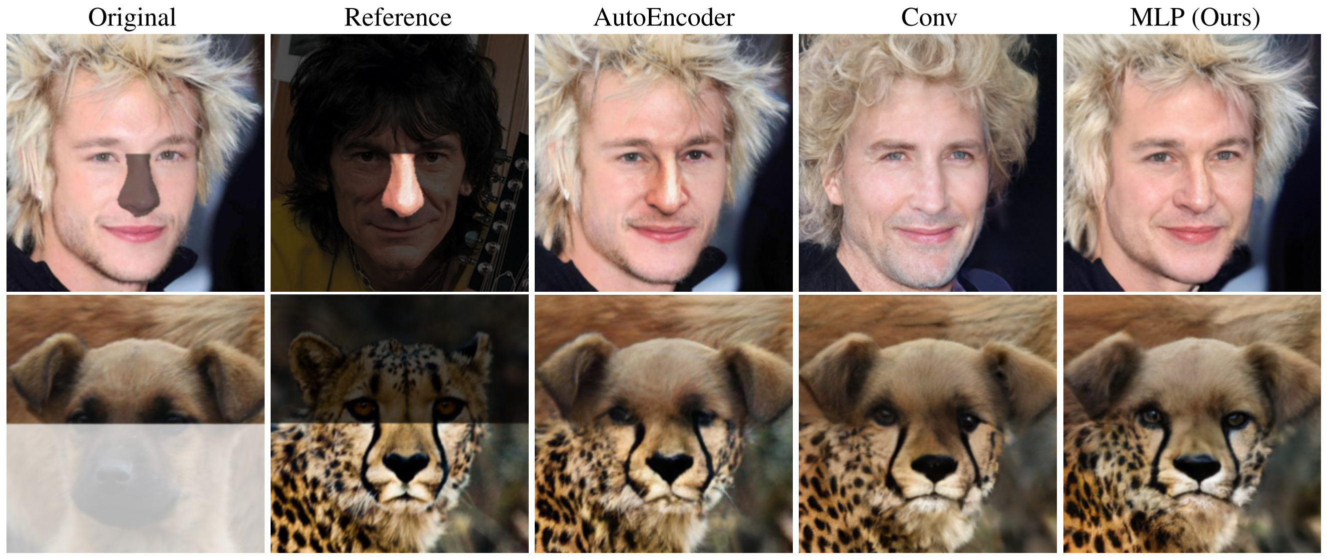


Figure 2: Local editing comparison across different mapping network architectures of StyleMapGAN. MLP-based architecture provides more natural images compared to autoencoder-based and convolution-based architecture.

training and higher performance than separately training the adversarial networks and the encoder, as described in §B. For the rest, we mostly follow the settings of StyleGAN2, *e.g.*, the discriminator architecture, R1 regularization [17] in the discriminator using $\gamma = 10$, Adam [13] optimizer with 0.002 learning rate, $\beta_1 = 0.0$ and $\beta_2 = 0.99$, an exponential moving average of the generator and the encoder, leaky ReLU [15], equalized learning rate [10] for all layers, random horizontal flip for augmentation, and reducing the learning rate by two orders [11] of magnitude for the mapping network. Our code is based on unofficial PyTorch implementation of StyleGAN2[1]. All StyleMapGAN variants at 256×256 are trained for two weeks on 5M images with 2 Tesla V100 GPUs using a minibatch size of 16. In §B, 1024×1024 models are trained for one week on 2.5M images with 8 Tesla V100 GPUs using a minibatch size of 16. We note that most cases keep slowly improving until 10M images. Our code is publicly available online for reproducibility[2].

Mapping network design for stylemap. There are several choices when designing a mapping network. We can remove the mapping network so that our method does not generate images from the standard Gaussian distribution and uses only real images for training like autoencoder [8]. As shown in Figure 2, autoencoder fails to produce realistic images using the projected stylemap. It seems to copy and paste between two images on RGB space. The autoencoder uses only images as input, which is a discrete variable. On the contrary, our method uses not only images but also the latent from Gaussian distribution, which is a continuous space. If we mix two latent codes for editing the image, training with continuous latent space can cover more latent values than discrete latent space.

Alternatively, we can easily think of convolutional layers due to the spatial dimensions of the stylemap. But, the mapping network with convolutional layers struggles in reconstruction so that the edited results images are quite different from the original images. We assume that there is such a limit because the convolutional layer's mapping is bounded to the local area. On the other hand, each weight and input in MLP are fully-connected so that it can make a more flexible latent space.

D. Loss details

In Section 3.2 of the main paper, we briefly introduced six losses. In this section, we provide details of the losses and their responsibilities. Some losses degrade reconstruction quality (MSE, LPIPS [26]), but we need every loss for the best editing quality (FID_{lerp}). We can obtain the best FID_{lerp} by training with all losses. Table 2 shows the quantitative results of the ablation study. The coefficients of all loss terms are set to 1.

Adversarial loss. The discriminator tries to classify fake images as fake, which are generated randomly from Gaussian distribution or reconstruction of input images. On the contrary, the generator fools the discriminator by producing more realistic images. Generation from the continuous

[1]https://github.com/rosinality/stylegan2-pytorch
[2]https://github.com/naver-ai/StyleMapGAN

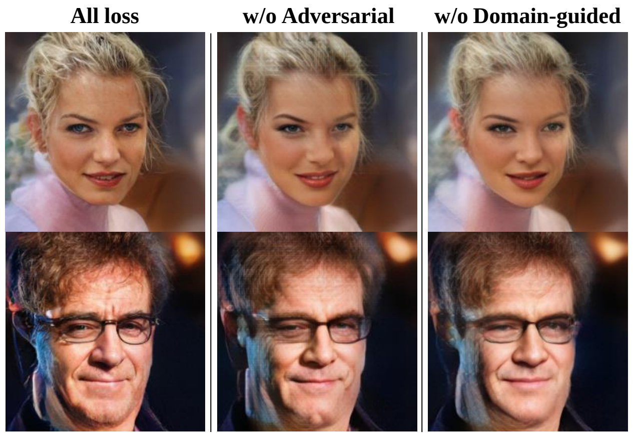


Figure 3: This figure shows the interpolation results of different networks. Leftmost images are results of a network trained using whole losses. Second column images are generated by a network trained without random Gaussian noise, which is similar to AutoEncoder. A network trained without domain-guided loss generates rightmost column images.

Removed loss	MSE	LPIPS	FID	FID_{lerp}
Adversarial loss	**0.009**	**0.137**	278.87	12.99
Domain-guided loss	0.013	0.193	5.11	16.84
Latent reconstruction loss	0.021	0.220	**4.43**	10.08
Image reconstruction loss	0.029	0.254	5.01	10.29
Perceptual loss	0.033	0.304	5.34	13.33
R1 regularization	0.097	0.403	31.82	14.56
Train with all losses	0.023	0.237	4.72	**9.97**

Table 2: Loss ablation study removing one loss at a time. We used CelebA-HQ, 256×256 image, and 8×8 stylemap.

space increases generation power in terms of smooth interpolation. Without adversarial loss related to the mapping network, we can not obtain a smooth manifold of latent space as mentioned in §C. Figure 3 also shows unnatural interpolation results and checkerboard artifacts if we don't use adversarial loss. We use the non-saturating loss [7] as our adversarial loss.

Domain-guided loss. Domain-guided loss is introduced by In-DomainGAN [27]. We use an adversarial training manner on fake images generated from real images via the encoder and the generator. The discriminator tries to classify generated images as fake while the encoder and the generator attempt to fool the discriminator. The loss pushes the projected latent code to remain in the original latent space of GAN, which facilitates smooth real-image editing by exploiting GAN's properties (*e.g.*, smooth interpolation). Without domain-guided loss, interpolation results are blurry as shown in Figure 3.

Latent reconstruction loss. The goal of the encoder is to find the latent code which generates the target image. When we generate a fake image from Gaussian distribution, we know the pair of the latent code and the generated image. Using that supervision, we train the encoder like other approaches [24, 23, 19, 27]. The encoder tries to project images in the semantic domain of the original latent space and alleviates strong bias against pixel-level reconstruction.

Image reconstruction loss. To make the output image visually identical to the input image, we minimize differences between them at pixel-level. If we do not use this loss function, visual reconstruction fails as ALAE [19] does.

Perceptual loss. Image reconstruction loss often makes the encoder overfit and output blurry images. Several approaches [1, 2, 27] adopt perceptual loss [9], which exploits the features extracted by VGG [22], for perceptual-level reconstruction. We use LPIPS [26] for perceptual loss, which has better feature representation.

R1 regularization. R1 regularization [17] makes training stable. We find that lazy regularization [12] is enough and apply it every 16 steps for the discriminator. Without this loss function, performance degrades in all metrics.

E. Additional results

In this section, we show extensive qualitative results. §E.1 illustrates randomly generated images to show that the generation capability of our method does not degenerate compared to the baseline. §E.2 and §E.3 provide expanded comparison on reconstruction and local editing, respectively. §E.4 shows additional unaligned transplantation examples. Our method is applicable to other latent-based editing methods as shown in §E.5 and §E.6. Lastly, we discuss the limitations (§E.7) of our method.

E.1. Random generation

The primary objective of GANs is generating high-fidelity images from random Gaussian noise. We show random generation results for each dataset: CelebA-HQ [10], AFHQ [5], and LSUN Car & Church [25]. We use 8×8 resolution of stylemap except for AFHQ, in which case 16×16 resolution provides much better generation quality as shown in Table 2 of the main paper. To generate high-quality images, we use the truncation trick [4, 14, 16] with $\psi = 0.7$. Figure 4 shows uncurated images and FID values. In CelebA-HQ and AFHQ, we use the same FID protocol as in the main experiments; the number of generated samples equal to that of training samples. On the other hand, LSUN consists of a lot of training images so that we use 50k images randomly chosen from the training set; the number of generated samples is 50k, too. Low FIDs reveal that our method has satisfactory generation capability.

E.2. Image projection & Interpolation

Although encoder-based methods project images into latent space in real time, their projection quality falls short of expectations. Figure 5 shows the projection quality comparison between our method and other encoder-based baselines (ALAE [19], In-DomainGAN [27], and SEAN [28]).

Figure 6 shows projection and interpolation results of our method and Image2StyleGAN [1]. Although Image2StyleGAN reconstructs the input images in high-fidelity, it struggles in latent interpolation because its projection on $\mathbf{w}^+$ drifts from the learned latent space $\mathbf{w}$ of the generator.

E.3. Local editing

Figure 7 shows local editing comparison with competitors. We eject two competitors (Structured Noise [3] and Editing in Style [6]) due to their poor results, as shown in Figure 4 of the main paper. It is because they do not target editing real images but target fake images.

E.4. Unaligned transplantation

Figure 8 and 9 show unaligned transplantation results in LSUN Car & Church [25]. Our method can transplant the arbitrary number and location of areas in reference images to the original images. Note that our method adjusts the color tone and structure of the same reference regarding the original images.

E.5. Semantic manipulation

We exploit InterFaceGAN [21] to find the semantic boundary in the latent space. Our method can change the semantic attribute using a certain direction derived from the boundary. We apply the direction on stylemap ($\mathbf{w}$ space). Figure 10 shows two versions of semantic manipulation. The global version is a typical way to manipulate attributes. The local version is only available in our method due to the spatial dimensions of the stylemap. We apply the semantic direction on the specified location in the $\mathbf{w}$ space. It allows us not to change the undesired area of the original image regardless of attribute correlation. For example, "Rosy Cheeks" makes lips red and "Goatee" changes the color of noses in the global version but not in the local version as shown in Figure 10. Furthermore, we can change part of attributes such as lip makeup from "Heavy Makeup" and beard from "Goatee". It alleviates the hard labor of highly granular labeling. Swap Autoencoder [18] shows region editing that the structure code also can be manipulated locally. However, it can not apply region editing on some attributes (*e.g.*, "Pale Skin") which are related to color and textures due to the absence of spatial dimensions in the texture code.

E.6. Style mixing

StyleGAN [11] proposed the style mixing method, which copies a specified subset of styles from the reference image. We operate style mixing in resized stylemaps ($\mathbf{w}^+$). Unlike StyleGAN, our generator has color and texture information in the resized stylemaps of low resolution (8×8). On the other hand, it generates overall structure through other resolutions ($16^2 - 256^2$). If we want to bring the color and texture styles from the reference image, we replace 8×8 resized stylemaps by reference. Figure 11 shows the examples.

Using style mixing and unaligned transplantation, we can transfer local structure only as shown in Figure 12. We use the original image on the first resized stylemap and the reference image for the remaining resolutions in the target region.

E.7. Failure cases

Our method has a limitation when original and reference images have different poses and target semantic sizes. Figure 13 shows failure cases on different poses. Especially, hair is not interpolated smoothly. Figure 14 shows failure cases on different target semantic sizes. The sizes and poses of the bumper vary, and our method can not transplant it naturally. This limitation gets worse when the resolution of the stylemap increases. Resolving this problem would be interesting future work.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *CVPR*, 2019. 4, 5

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *CVPR*, 2020. 4

[3] Yazeed Alharbi and Peter Wonka. Disentangled image generation through structured noise injection. In *CVPR*, 2020. 5

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. 4

[5] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 4

[6] Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. Editing in style: Uncovering the local semantics of gans. In *CVPR*, 2020. 5

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville,

and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014. 4

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 2006. 3

[9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 4

[10] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2018. 3, 4

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3, 5

[12] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 2, 4

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[14] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. 2018. 4

[15] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 3

[16] M. Marchesi. Megapixel size image creation using generative adversarial networks. *ArXiv*, abs/1706.00082, 2017. 4

[17] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *ICML*, 2018. 3, 4

[18] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In *NeurIPS*, 2020. 5

[19] Stanislav Pidhorskyi, Donald Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020. 4, 5, 8

[20] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. 2016. 2

[21] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020. 5

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4

[23] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NeurIPS*, 2017. 4

[24] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *AAAI*, 2017. 4

[25] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 4, 5

[26] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 3, 4

[27] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, 2020. 4, 5, 8, 10

[28] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *CVPR*, 2020. 5, 8, 10

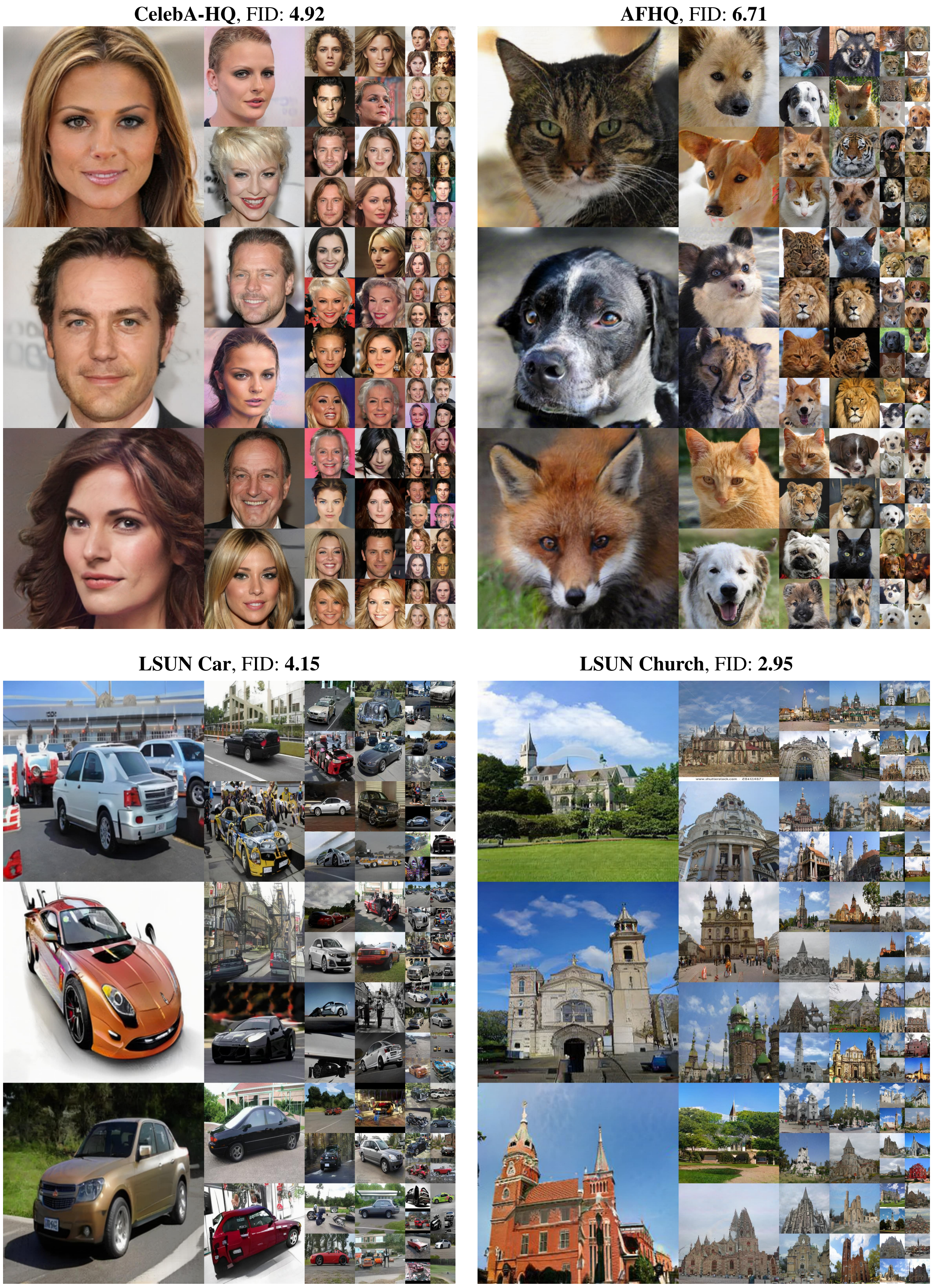

CelebA-HQ, FID: **4.92**
AFHQ, FID: **6.71**
LSUN Car, FID: **4.15**
LSUN Church, FID: **2.95**

Figure 4: Uncurated random generation results in four datasets.

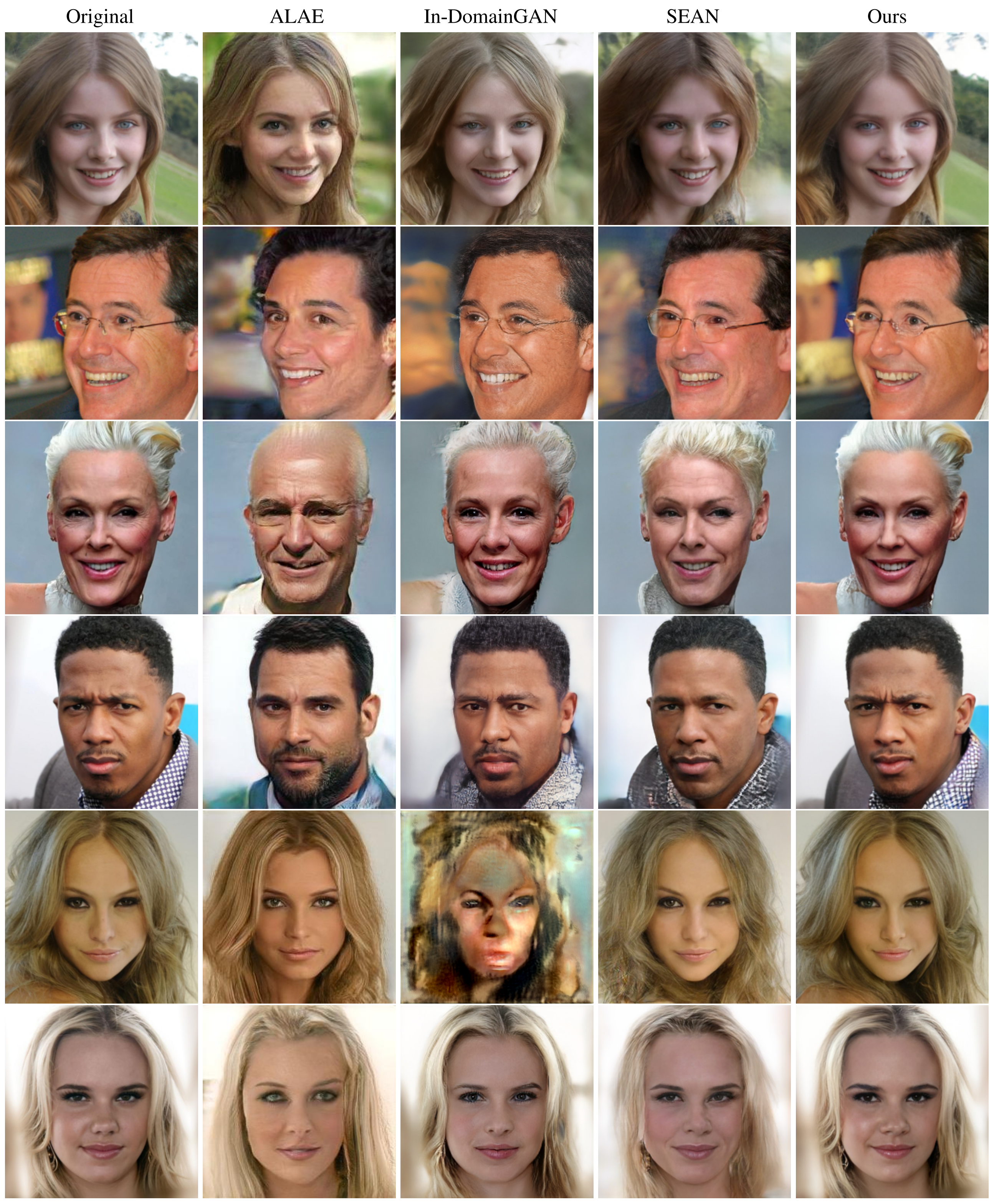


Figure 5: Reconstruction results in encoder-based methods. ALAE [19] does not preserve identities in the original images. In-DomainGAN [27] has better reconstruction results than ALAE, but it sometimes fails to generate human-like images as shown in the second-last row. Note that In-DomainGAN requires additional optimization steps which take seconds. SEAN [28] fails to preserve the shape of original images (*e.g.*, background, hair curl, and cloth). Our method reconstructs images well not only the color and texture but also the shape.

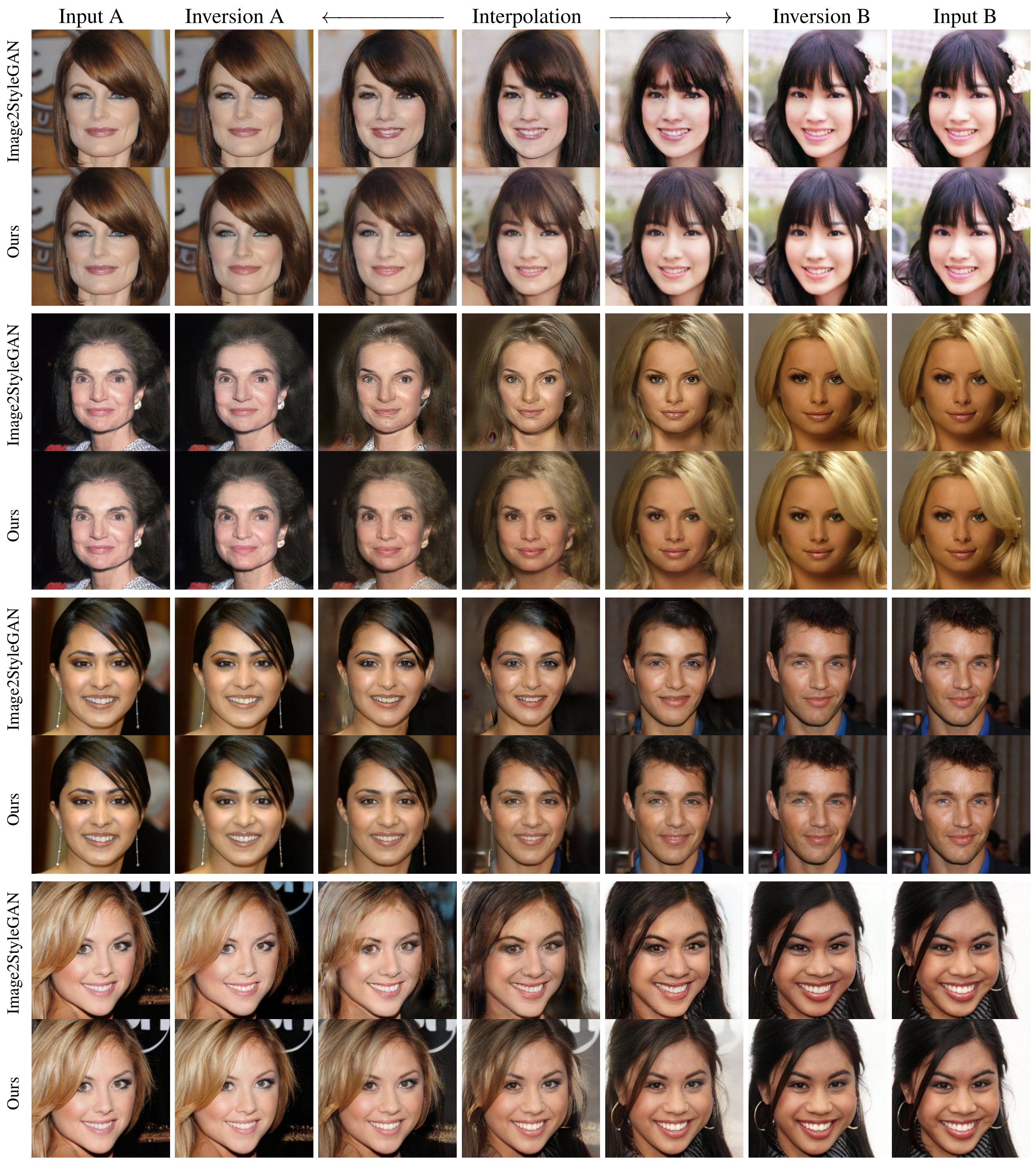


Figure 6: Comparison with Image2StyleGAN for interpolation quality. Image2StyleGAN shows rugged and discontinuous interpolations, even though the reconstruction quality is visually good. Our method produces clearer and smoother interpolations, which reveal our superiority in both pixel-level and semantic-level (*i.e.*, the semantics the original latent space) reconstruction. Note that the reconstruction speed of our method is 2000$\times$ faster than Image2StyleGAN.

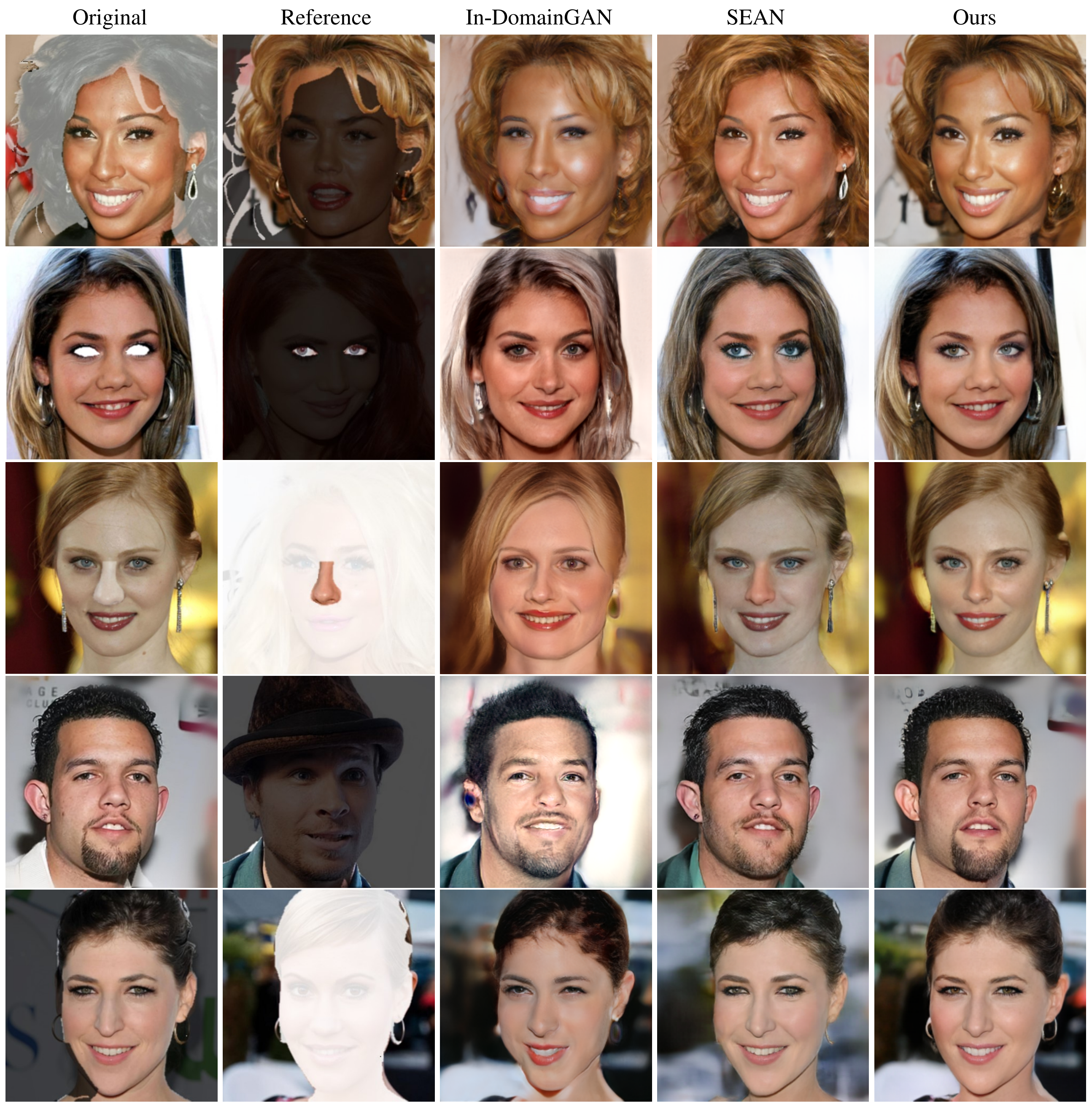


Figure 7: Local editing comparison in CelebA-HQ. Each row edits hair, eyes, nose, cloth, and background. In-DomainGAN [27] only optimizes region in the target mask and it changes identities of the original images. SEAN [28] tends to bring only the color and the texture of reference images, not the shape (especially on hair lines). Our method reflects the shape of the reference image as well and preserves the identity of the original image.

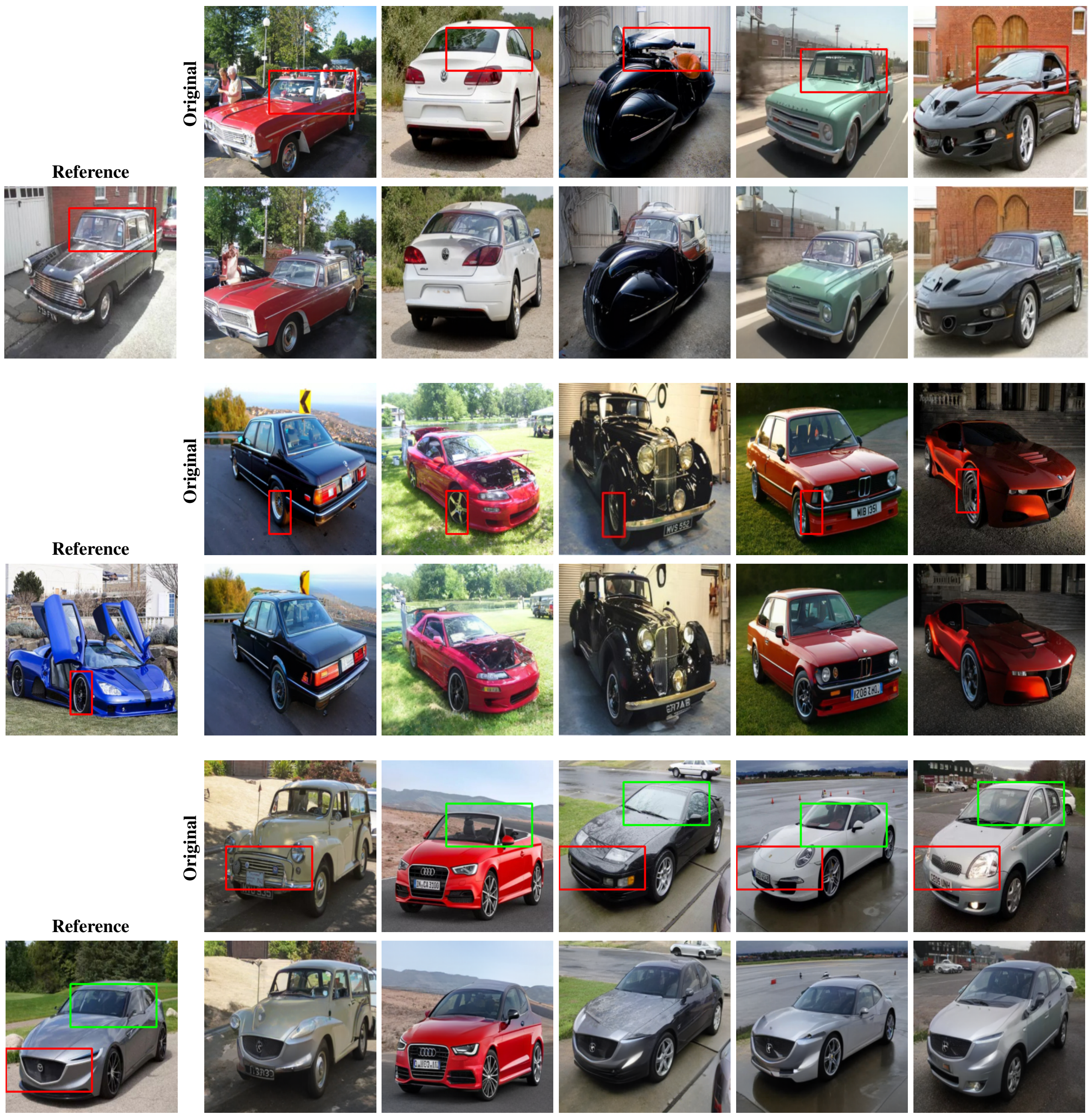


Figure 8: Transplantation results of our method in LSUN Car. We transplant the cabin, wheel, and bumper.

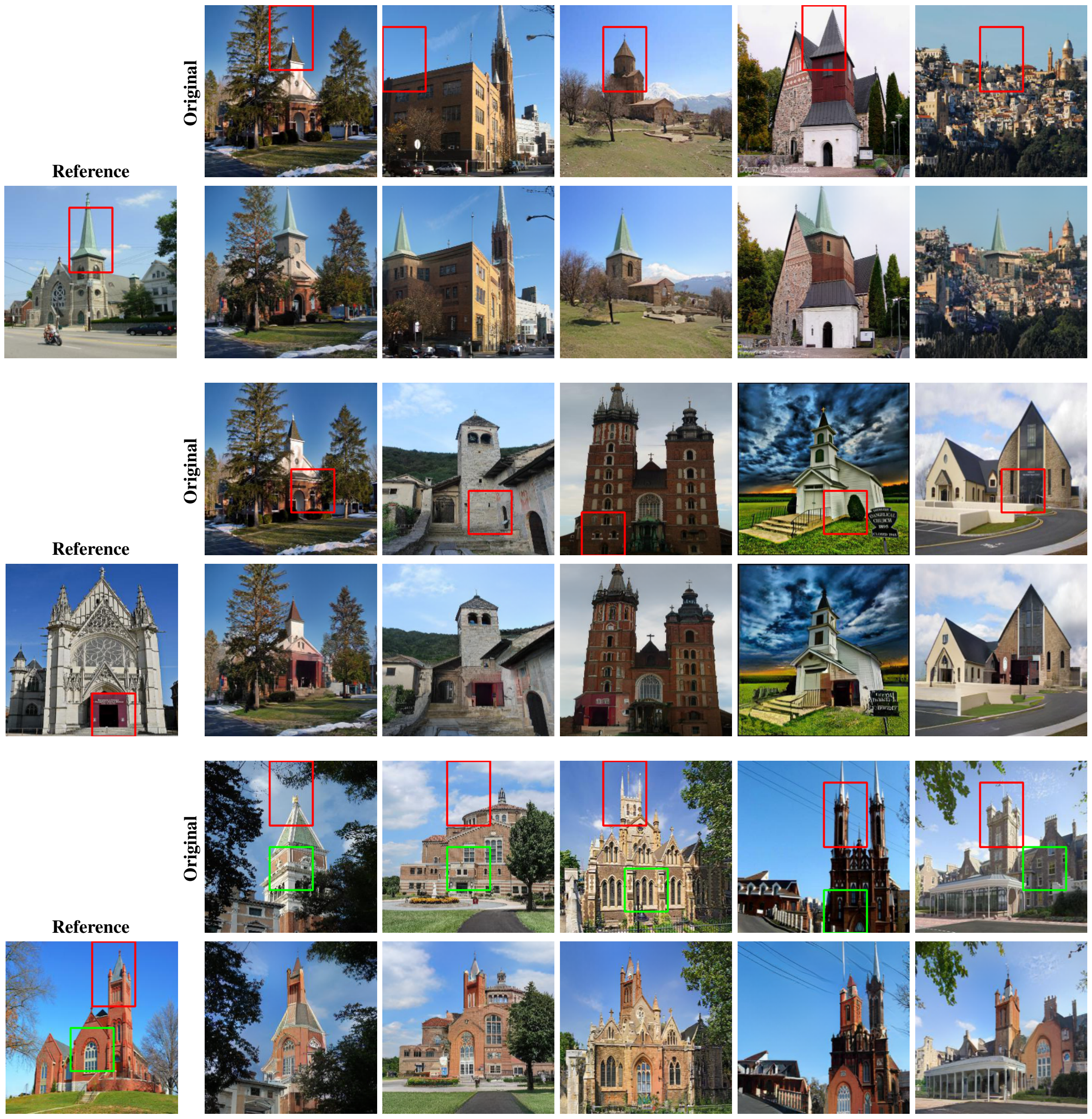


Figure 9: Transplanting tower, gate and windows in LSUN Church by our method. Our method can transplant the arbitrary number and location of areas in reference images to the original images. Note that our method adjusts color tone and structure of the same references regarding the original images.

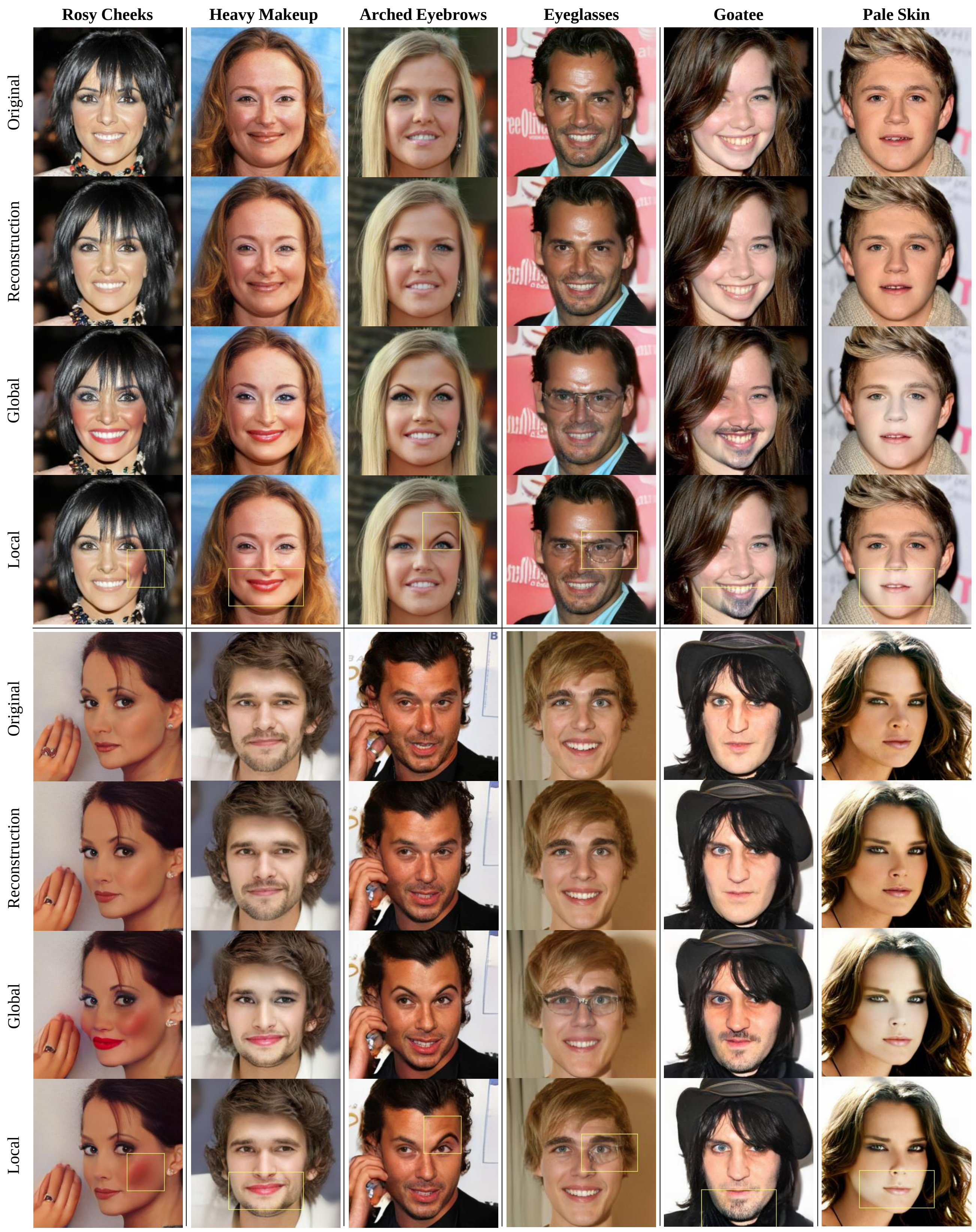


Figure 10: The results of the global and local version of semantic manipulation in our method. In local manipulation, we apply the semantic direction in the yellow box area.

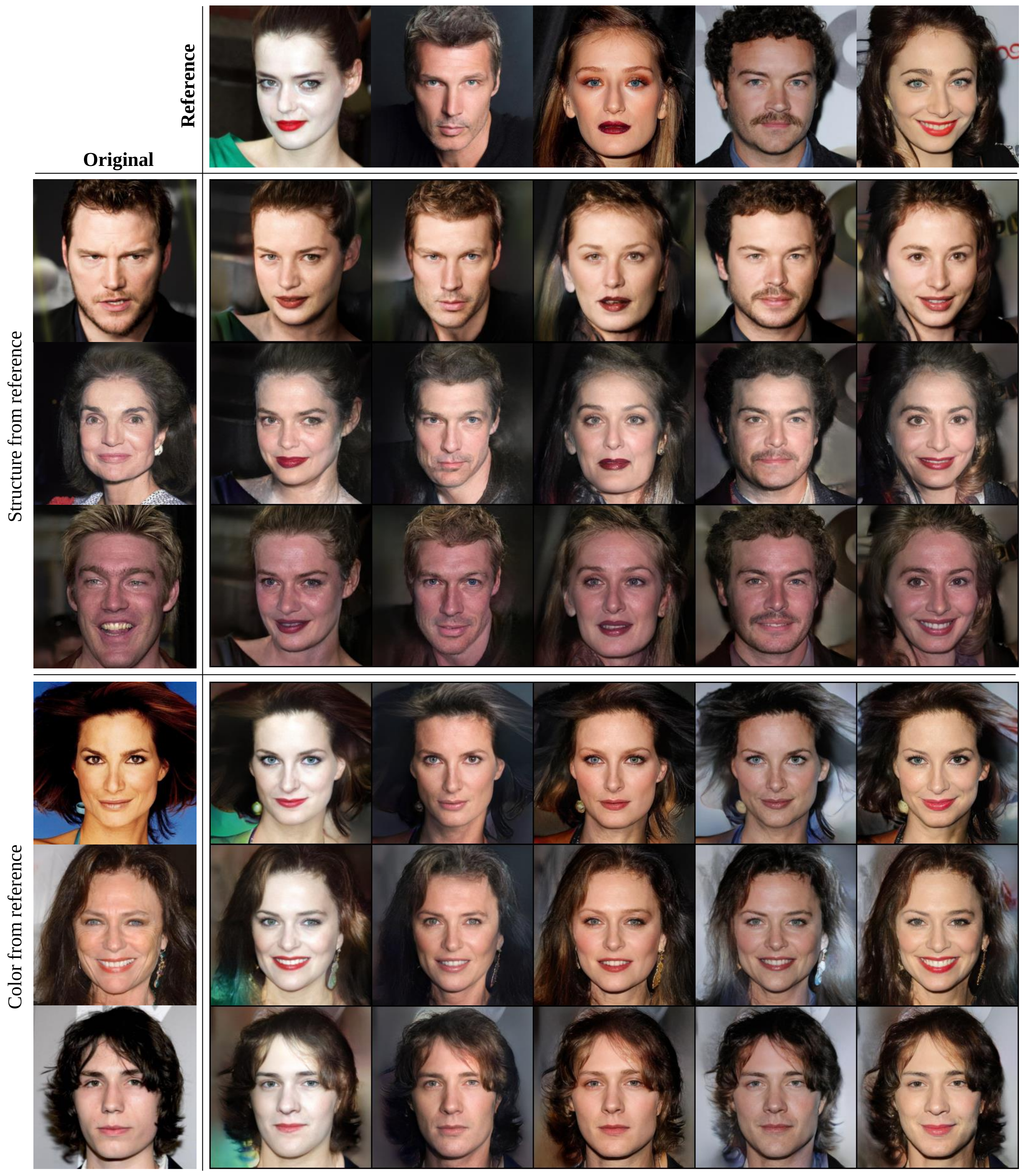


Figure 11: The results of style mixing in our method. Please refer to E.6 for details.

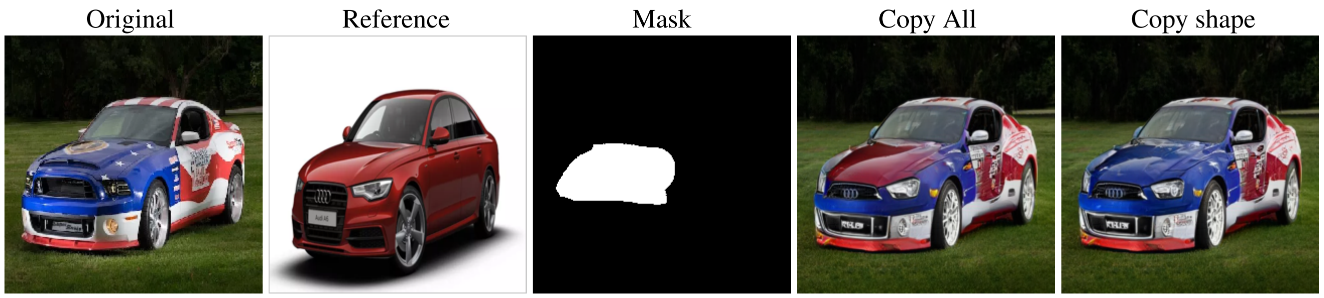


Figure 12: 4th column shows the transplantation of all identity including shape, texture, and color. The rightmost image shows the transplantation of structure alone.

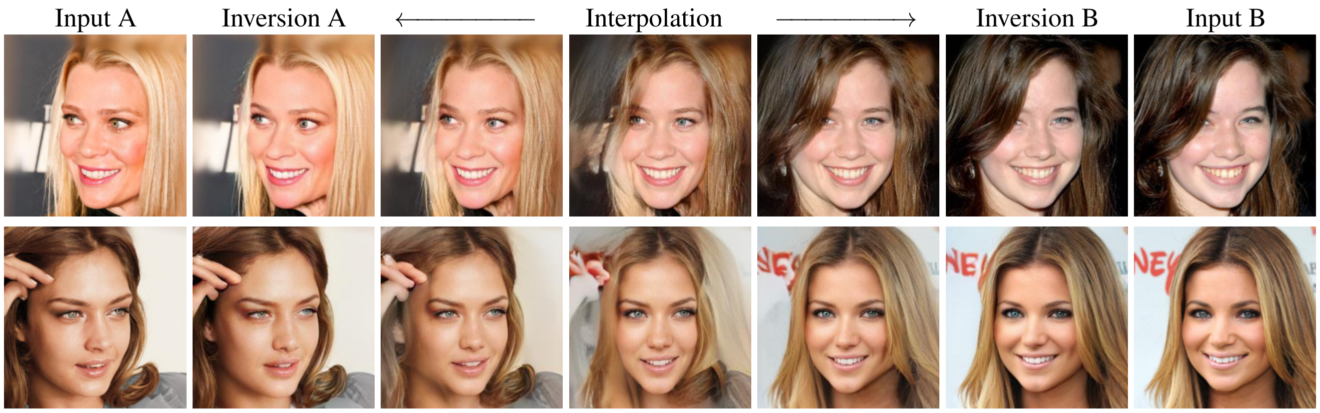


Figure 13: Failure cases of interpolation in our method due to extreme pose difference.

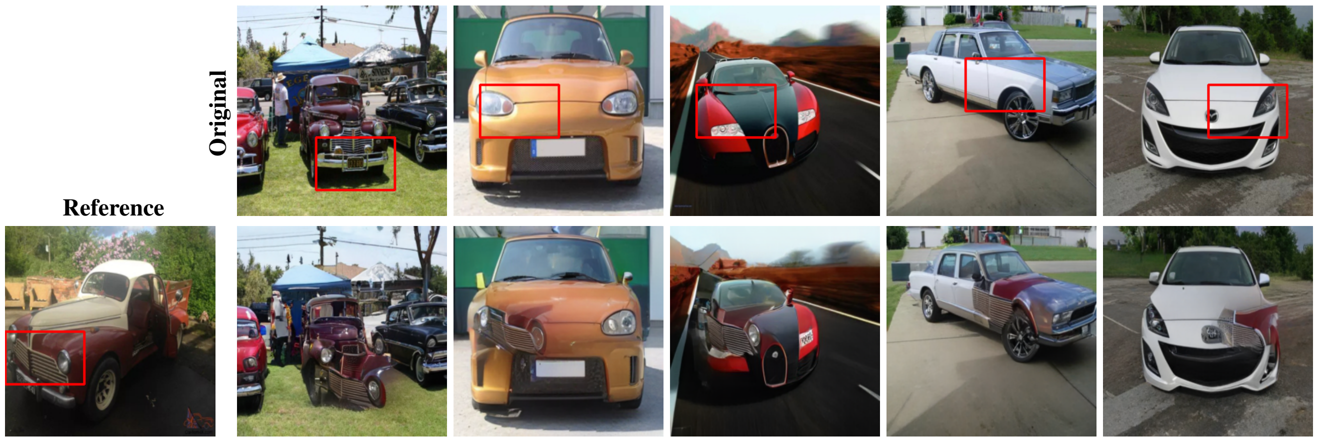


Figure 14: Failure cases of transplantation in our method due to different sizes of the masks.