GAN with Multivariate Disentangling for Controllable Hair Editing

Xuyang Guo[1,2], Meina Kan[1,2], Tianle Chen[1,2], and Shiguang Shan[1,2,3]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
{xuyang.guo,tianle.chen}@vipl.ict.ac.cn, {kanmeina,sgshan}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beiijng 100049, China
[3] Peng Cheng Laboratory, Shenzhen, 518055, China

Abstract. Hair editing is an essential but challenging task in portrait editing considering the complex geometry and material of hair. Existing methods have achieved promising results by editing through a reference photo, user-painted mask, or guiding strokes. However, when a user provides no reference photo or hardly paints a desirable mask, these works fail to edit. Going a further step, we propose an efficiently controllable method that can provide a set of sliding bars to do continuous and fine hair editing. Meanwhile, it also naturally supports discrete editing through a reference photo and user-painted mask. Specifically, we propose a generative adversarial network with a multivariate Gaussian disentangling module. Firstly, an encoder disentangles the hair's major attributes, including color, texture, and shape, to separate latent representations. The latent representation of each attribute is modeled as a standard multivariate Gaussian distribution, to make each dimension of an attribute be changed continuously and finely. Benefiting from the Gaussian distribution, any manual editing including sliding a bar, providing a reference photo, and painting a mask can be easily made, which is flexible and friendly for users to interact with. Finally, with changed latent representations, the decoder outputs a portrait with the edited hair. Experiments show that our method can edit each attribute's dimension continuously and separately. Besides, when editing through reference images and painted masks like existing methods, our method achieves comparable results in terms of FID and visualization. Codes can be found at https://github.com/XuyangGuo/CtrlHair.

Keywords: Controllable Editing, Hair Editing, Style Transfer, Style Manipulation

1 Introduction

Hair consists of numerous, dedicated, and small strands, whose complex geometry and material lead to various hair appearances in color, structure, shape, style, etc. The complex nature of hair makes it difficult to model, depict and generate. Hair editing, as a kind of image manipulation, is one of the most challenging and important components of portrait editing. The task of hair editing

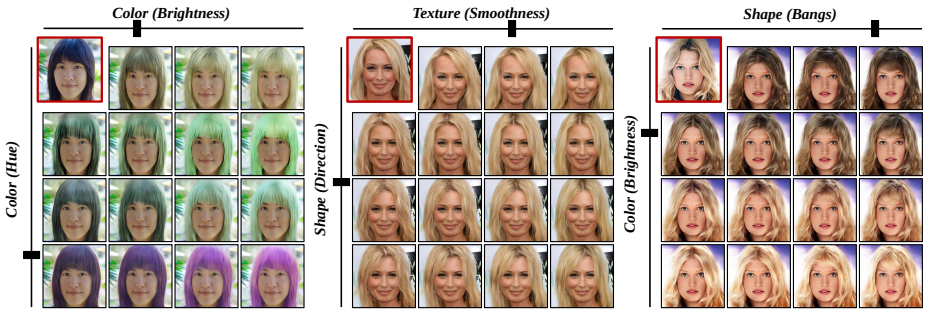


Fig. 1. Controllable hair editing by using CtrlHair. Given a portrait in the upper left corner of each subfigure, CtrlHair can edit the hair by sliding a set of bars. CtrlHair supports editing of fine-grained factors of an attribute simultaneously and separately

has valuable application scenarios, such as animation, games design, and virtual reality, and can also help to better understand the generation of face images.

The development of Generative Adversarial Nets [6] has greatly boosted the hairstyle transfer [2,24,21,28], i.e. transferring the hairstyle of a reference image to a target image including shape, appearance, etc. These methods especially MichiGAN [24] achieve impressive results, which divides the hair into attributes, including appearance, structure, and shape, and then edits hair according to painted mask, guiding strokes, or reference photos for better user controllability. However, when a user has no reference photo or hardly paints a desirable mask, these existing works fail to edit. Go a further step, our work endeavors to provide an efficiently controllable hair editing method, named as *CtrlHair*, with which a user can do arbitrary hair editing by simply sliding a set of control bars, providing reference photos, or painting a mask. Besides, rather than transferring or editing an entire attribute such as color, shape, texture, we want to explore manipulating each dimension of an attribute for finer control, e.g., change the direction of hair shape and the brightness of hair color, as shown in Fig. 1.

Specifically, we propose a GAN with a multivariate Gaussian disentangling module. Firstly, an encoder disentangles the hair's major attributes, including color, texture, and shape, to separate latent representations, one for each attribute. The latent representation of each attribute is modeled as a standard multivariate Gaussian distribution, to make each dimension can be changed *continuously* and *finely*. Benefiting from the Gaussian distribution, any manual editing including sliding a bar, providing a reference photo, painting a mask can be easily made, which is flexible for users to interact with. Finally, with changed latent representations, a decoder outputs an image with edited hair.

Among all attributes, shape editing not only change the hair region but also affects other parts of the portrait. To avoid the problem of tricky misalignment between hair and face region and inpainting caused by shape change, we especially propose a novel Shape Adaptor module. The shape adaptor takes the

changed hair shape, face mask, and background mask as input, and automatically does alignment and inpainting to output an appropriate portrait mask including hair region, face region, and background region.

Briefly, our contributions are summarized in three-folds:

1. We propose a GAN with multivariate Gaussian disentangling, with which a user can do continuous and fine-grained hair editing by simply sliding a set of control bars, providing reference photos, or painting a mask.
2. Especially, for a realistic portrait with edited hair, a learning-based shape adaptor is proposed to automatically do alignment between hair and face region, as well as inpaint those uncovered regions caused by shape change.
3. Experiments show that our method can edit each dimension of each attribute continuously and separately. It also achieves comparable results as existing methods on task of hairstyle transfer.

2 Related Works

Image Generation. In recent years, attributing to the development of Generative Adversarial Networks [6], the realism and resolution of image generation have been significantly improved. The early method DCGAN [20] introduces a deconvolutional structure, which greatly improves realism. ProgressiveGAN [10] and StyleGANs [11,12] further propose a progressive generative network to achieve photo-realistic face images in high resolution. In addition, another kind of work focuses on conditional generation with supervised or unsupervised condition. The supervised conditional GAN methods, such as CGAN [15] and ACGAN [16], are trained with a label as a conditional signal to generate an image belonging to the condition-indicated class. Moreover, InfoGAN [3] learns disentangled representations in an unsupervised manner, by maximizing mutual information between conditional signal and generated images. Furthermore, conditional image generation is widely explored for different types of tasks, e.g., style transferring [9], generating images from segmentation masks [18], etc.

Face Editing. The above methods can generate images, but cannot edit a given image. So, there appear a few works using GAN to do image editing, especially face editing. Given an input image, these methods aim to generate an image where only one or several attributes are edited while the rest contents keep unchanged. Most face editing methods [19,22,4,27,14,7,5,17,23] manipulate specific face attributes by using binary attribute labels (e.g., with or without glasses/smiling). However, editing with binary attribute labels can only do coarsely editing. So, some works such as MaskGAN [13] are proposed to use facial segmentation masks to edit the face more flexibly and meticulously. This method can edit the shape or transfer the appearance style of the entire image. Different from encoding the appearance style of an image into one feature representation in MaskGAN [13], SEAN [29] further disentangles the appearance style of the entire image to multiple separate feature representations according to the segmentation masks, allowing more flexible editing.

Hair Editing. Among all face parts, hair has complex structures, shapes, appearances, etc, leading to many semantics and making fine annotations lacking. Hence, the general face editing methods mentioned above are hardly used for hair editing with meaningful semantic variation. So there are some works [2,24,21,28] that are specially dedicated to editing hair especially. Since hair is hardly labeled by discrete category for training like face editing, existing methods directly transfer the hair of a reference image to another. Chai et al. [2] and LOHO [21] disentangle the attributes of hair into two parts, including shape-relevant attributes and shape-irrelevant attributes, which can be separately transferred from a reference image to another image. Moreover, MichiGAN [24] and Barbershop [28] further disentangle hair into three orthogonal attributes, including shape, structure, and appearance, and users can edit these attributes by providing reference images, painted masks, or guiding strokes. Overall, these methods can edit or transfer the hairstyle by reference photos or painted masks as conditions. However, when a user has no reference photos or hardly paints a desirable mask, these works fail to edit. That is what our work attempt to deal with.

3 CtrlHair

Given an input portrait I, our CtrlHair attempts to edit the attributes of hair region continuously and finely. The overall editing process is as follows:

$$(X, S) = \Phi(I), \quad (\hat{X}, \hat{S}) = \text{CtrlHair}(X, S), \quad \hat{I} = \Psi(\hat{X}, \hat{S}, I), \tag{1}$$

where Φ parses the input portrait I to get a segmentation mask S, including hair mask S_H, face mask S_F and background mask S_B, i.e. $S = (S_H, S_F, S_B)$. X can be just the hair region of the original I or can be a basic feature of hair for a better feature presentation. Ψ blends the edited hair region and other regions including face and background to a portrait $\hat{I}$. Since our work mainly focuses on the hair editing, Φ and Ψ are directly obtained by using a pre-trained model, i.e. pre-trained BiSeNet [25,30] is used to obtain portrait segmentation mask S and SEAN [29] is used to obtain basic feature representation X and generate portrait with edited hair $\hat{X}$ and $\hat{S}$.

As stated above, the main goal of our method is to edit hair continuously and finely. Continuous editing means that users can edit a given hair to any other rational look, or a look specified by a reference photo and user-painted mask. Fine editing means that users can separately edit fine-grained variation factor of a hair attribute, such as length of shape, hue of color, etc. Besides, the edited hair region should be harmoniously blended with other regions including face and background to generate a realistic portrait.

For this goal, we propose a generative adversarial network with a multivariate Gaussian disentangling module. The proposed method consists of three successive modules, i.e. encoding module, interactive editing module, and decoding module as shown in Fig. 2. We first introduce the three modules in the editing procedure of CtrlHair, and then presents the training objectives in detail.

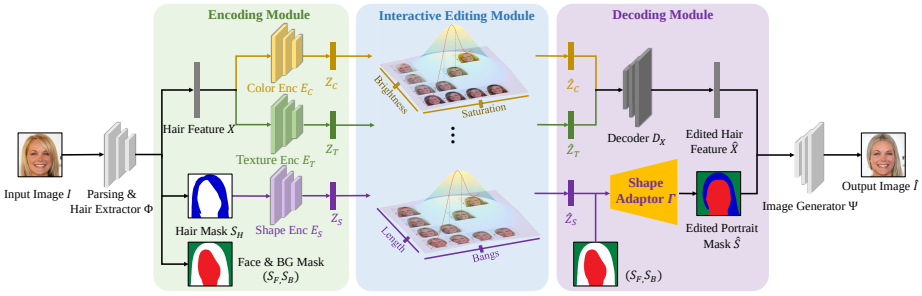


Fig. 2. An overview of CtrlHair, which consists of three modules: encoding, interactive editing, and decoding. The encoding module separates hair into distinct latent representations, of which each is formulated as a standard multivariate Gaussian distribution. In the interactive editing module, any manual editing including a set of bars, reference images, painted masks, can be made. The decoding module outputs an edited portrait with these edited latent representations

3.1 The Editing Procedure

Encoding Module. The encoding module aims at disentangling the hair into three latent representations, corresponding to color, texture, and shape. These latent representations are respectively obtained by three encoders as below:

$$Z_C = E_C(X), \quad Z_T = E_T(X), \quad Z_S = E_S(S_H), \tag{2}$$

where $Z_C, Z_T, Z_S \in \mathcal{N}(0, I)$ represent the disentangled representation of color, texture, and shape. E_C, E_T and E_S are corresponding encoders. For each attribute, expectedly it can be edited to any other look continuously and finely. Therefore, the representation of Z_C, Z_T and Z_S are all restricted to follow standard multivariate Gaussian distribution during training. Benefiting from the continuity of Gaussian distribution, any editing can be made by directly changing the value of Z_C, Z_T, and Z_S. Besides, all dimensions of standard multivariate Gaussian distribution are mutually independent, which makes that each dimension of an attribute can be separately edited for fine editing, such as length of shape, hue of color, etc. The meaning of each dimension of an attribute can be learned without supervision to implicitly mine variational factors (e.g. smoothness of texture), and can be also supervised by labels to explicitly learn meaning factors (e.g. brightness of color).

Interactive Editing Module. Given the disentangled latent representation Z_C, Z_T and Z_S, users can easily edit each dimension of each attribute, simply by sliding a set of bars, and get edited latent representation $\hat{Z}_C$, $\hat{Z}_T$ and $\hat{Z}_S$. Besides, it also naturally supports users to edit according to reference photos or painted masks like existing methods [24,21,28] by inputting them to the encoders to

get their latent representation. These processes can be illustrated as one unified interactive operation as below:

$$(Z_C, Z_T, Z_S) \xrightarrow{f_{\text{sliding bars}};\ f_{\text{references}};\ f_{\text{painted mask}}} (\hat{Z}_C, \hat{Z}_T, \hat{Z}_S). \tag{3}$$

Decoding Module. The decoding module aims at generating an edited image based on the modified latent representation. The appearance feature and shape of hair are generated respectively. Color and texture are both related to the appearance of hair, so they are decoded together via a single decoder as:

$$\hat{X} = D_X(\hat{Z_C}, \hat{Z_T}), \tag{4}$$

where D_X is the decoder for color and texture.

Similarly, the shape of hair can be also decoded from Z_S like appearance in Eq.(4). However, different from color and texture, shape editing not only changes hair region but also affects the face and background region. On one hand, the edited shape should be well aligned with the face region in appropriate scale and width, e.g. a thin face with a large-shaped hair looks unrealistic. On the other hand, shape change would uncover some regions that are originally covered by hair. These regions should be inpainted as face or background. Existing methods ignore these two problems, or just throw them to GAN, leading to awkward portrait or inaccurate hair shape compared with the reference image. Therefore, we specifically propose a learning-based shape adaptor Γ to obtain a well-aligned hair shape and well-inpainted portrait mask as below:

$$\hat{S} = \Gamma(\hat{Z_S}, S_F, S_B). \tag{5}$$

In the portrait mask $\hat{S}$, the hair shape is slightly adjusted, and the background mask and face mask are inpainted if they are uncovered after changing hair shape, to ensure a harmonious and realistic portrait image generated by Eq.(1).

3.2 The Training Objectives

Expectedly, the generated image from the whole editing process in Eq.(1) should satisfy three requirements, i.e. the edited image $\hat{I}$ should look realistic; the hair of $\hat{I}$ is correctly edited; continuous and fine editing is supported. These three characteristics are formulated as three types of objectives including realism loss $\mathcal{L}^{real}$, reconstruction loss $\mathcal{L}^{rec}$ and distribution loss $\mathcal{L}^{dist}$ as follows:

$$\mathcal{L} = \lambda^{real}\mathcal{L}^{real} + \sum\nolimits_{k\in\{C,T,S\}} (\lambda_k^{rec}\mathcal{L}_k^{rec} + \lambda_k^{dist}\mathcal{L}_k^{dist}). \tag{6}$$

Firstly, the generated image should be realistic. Since the Φ and Ψ are pre-trained, we only need to ensure the generated $\hat{X}$ and $\hat{S}$ same as those from the real images. So, a realism adversarial loss $\mathcal{L}^{real}$, i.e., the first term in Eq.(6), is optimized by adversarial training between generated $(\hat{X}, \hat{S})$ and (X, S) from real images, same as the adversarial training in conventional GANs. *Secondly*,

the editing should be correct, i.e. the generated image $\hat{I}$ should be with the edited attributes while other attributes remain unchanged. These are achieved by the reconstruction loss $\mathcal{L}^{rec}$, i.e. the second term in Eq.(6). *Thirdly*, the representation Z_C, Z_T, Z_S are expected to follow a standard multivariate Gaussian distribution for continuous and fine editing, which is formulated by distribution loss $\mathcal{L}^{dist}$, i.e., the third term in Eq.(6).

The reconstruction loss and distribution loss of each attribute are different respecting to their distinct natures, which are introduced respectively as follows.

Color. For color, the latent representation $Z_C \in \mathbb{R}^4$ is designed as a 4-dimensional vector. The first 3 dimensions represent the HSV color values, which capture the mean color of the hair region. The 4th dimension represents the variance considering the color of all pixels in a hair region is not identical but usually different. The reconstruction loss and distribution loss of color are formulated as below:

$$\min_{E_C} \mathcal{L}_C^{dist} = \mathbb{E}_{(X,\tilde{Z}_C)\sim P_d} \left\| E_C(X) - \tilde{Z}_C \right\|_2^2, \quad \tilde{Z}_C \sim \mathcal{N}(0, I) \tag{7}$$

$$\min_{D_X} \mathcal{L}_C^{rec} = \mathbb{E}_{\hat{Z}_C, \hat{Z}_T \sim \mathcal{N}(0,I)} \left\| E_C(\hat{X}) - \hat{Z}_C \right\|_2^2. \tag{8}$$

where P_d is the training set. Eq.(7) ensures that the latent representation generated from real images follows gaussian distribution through a supervised manner. Since H, S, V, and variance of the hair region can be easily calculated without manual labeling, they are used as the supervised color label. Specifically, the distribution of (H, S, V, variance) of hair region from training images is transformed to standard Gaussian distribution dimension-wisely, achieved by analytically mapping the quantiles of the cumulative distribution of training samples to standard Gaussian distribution, denoted as $\tilde{Z}_C$. Then Eq.(7) enforces the encoded latent representation to be the same as its supervised label $\tilde{Z}_C$, i.e. enforce $Z_C = E_C(X)$ follow standard multivariate Gaussian distribution.

Eq.(8) ensures that each $\hat{Z}_C$ sampled from $\mathcal{N}(0, I)$ can correctly manipulate the generated hair feature $\hat{X}$ via reconstruction between the sampled $\hat{Z}_C$ and encoded $E_C(\hat{X})$ from generated feature with it.

Besides color, any other supervised attributes can be easily added similarly.

Texture. The texture attribute captures the pattern and regularity of the hair, such as hair-strand thickness, smoothness, curliness, etc. These semantics are difficult to label manually, so we use unsupervised methods to discover them.

On one hand, the low dimensional latent representation Z_T should be capable of manipulating the image, formulated as two reconstruction loss as follows:

$$\min_{E_T, D_X} \mathcal{L}_T^{rec} = \mathbb{E}_{X\sim P_d} \| X - D_X(E_C(X), E_T(X)) \|_2^2 \tag{9}$$

$$+ \mathbb{E}_{\hat{Z}_C, \hat{Z}_T \sim \mathcal{N}(0,I)} \left\| E_T(\hat{X}) - \hat{Z}_T \right\|_2^2. \tag{10}$$

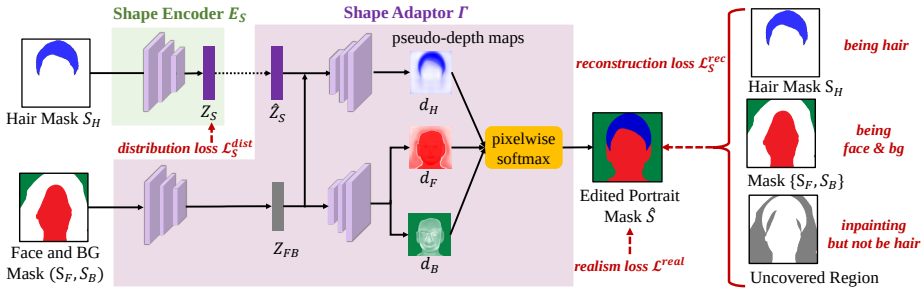


Fig. 3. The design of shape editing. The shape adaptor takes latent representation $\hat{Z}_S$ of edited hair shape, original face and background mask S_F, S_B as input; then adjusts them to obtain an edited portrait mask, whose hair region, inpainting region, and the rest face and background region are constrained accordingly

The reconstruction in Eq.(10) maximizes the mutual information [3] between $\hat{Z}_T$ and $\hat{X}$, which ensures each dimension of $\hat{Z}_T$ can manipulate some kind of variation in $\hat{X}$. The reconstruction in Eq.(9) enforces $E_T(X)$ to try to capture a meaningful variation of texture rather than other attributes or a trivial solution.

On the other hand, Z_T should follow standard multivariate Gaussian distribution. This is partly achieved by reconstructing sampled $\hat{Z}_T$ in above Eq.(10). Besides, an auxiliary constraint is used to directly enforce the low-order moments of $\hat{Z}_T$ to be same as that of standard Gaussian distribution. Here, one- and second-order moments (i.e. $\mu = \mathbf{0}$, and $\Sigma = \mathbf{I}$) are adopted, formulated as:

$$\min_{E_T, D_X} \mathcal{L}_T^{dist} = \|\mathbb{E}_{X\sim P_d} E_T(X) - \mu\|_2^2 + \|\mathbb{V}_{X\sim P_d} E_T(X) - \Sigma\|_F^2. \tag{11}$$

Shape. Shape is a complex attribute since shape not only changes the hair region but also affects the face and background region. Therefore, the shape editing should be not only correct but also reasonable, i.e. hair shape should be well aligned with the face region and the uncovered region should be inptained reasonably. Hence, a shape adaptor Γ is proposed to simultaneously adjust the shape to align with the face and also inpaint those uncovered regions.

To achieve it, during training, we conduct constraints by sampling a person's hair mask $S_H \in \{0,1\}^{H\times W\times 1}$, and adapt it to another person with his face mask and background mask $(S_F, S_B) \in \{0,1\}^{H\times W\times 2}$. The objective loss is calculated on the entire generated portrait mask $\hat{S}$ instead of only on the hair region. As shown in Fig. 3, the shape adaptor takes edited shape representation $\hat{Z}_S$, face and background mask (S_F, S_B) as input, and then generates the pseudo depth-map of hair, face, and background, denoted as $d_H, d_F, d_B \in \mathbb{R}^{H\times W\times 1}$. The pseudo depth-maps allow easy and soft composition [1] of them to get the adjusted portrait mask $\hat{S}_k = \frac{\exp(d_k)}{\sum_{m\in\{H,F,B\}}\exp(d_m)}$ $(k \in \{H, F, B\})$.

Table 1. Functionality comparison of different methods

Functionality	MichiGAN [24]	LOHO [21]	BarberShop [28]	CtrlHair (ours)
Interaction Mode	references painted mask sketch	references	references painted mask	references painted mask sliding bars
Editing Flexiblity	coarse, discrete			fine-grained, continuous
Shape Editing	replace directly			shape adaptor

The shape reconstruction loss is calculated on the adjusted portrait mask $\hat{S}$ from the shape adaptor as below:

$$\min_{E_S,\Gamma} \mathcal{L}_S^{rec} = -\mathbb{E}_{S_H,S_F,S_B\sim P_d}\Big[S_H \log(\hat{S}_H) + (1-S_H)\sum_{k\in\{F,B\}} S_k \log(\hat{S}_k) \quad (12)$$

$$+ (1-S_H)\log(1-\hat{S}_H)\Big], \quad (13)$$

where the first term in Eq.(12) enforces that the generated hair shape should be almost the same as the input while allowing slightly adjustment for different (S_F, S_B) guided by adversarial realism loss $\mathcal{L}^{real}$. The second term in Eq.(12) ensures that the unaffected face and background region remain unchanged. The unconvered region is determined automatically via adversarial realism loss $\mathcal{L}^{real}$, but with an extra constraint in Eq.(13) to prohibit being inpainted as hair again.

Besides, the distribution loss $\mathcal{L}_S^{dist}$ is designed as adversarial loss between the distribution of generated latent representation and Gaussian distribution:

$$\min_{E_S}\max_{\delta} \mathcal{L}_S^{dist} = \mathbb{E}_{S_H\sim P_d}[\log(1-\delta(E_S(S_H))] + \mathbb{E}_{Z_S^r\sim\mathcal{N}(0,I)}[\log\delta(Z_S^r)], \quad (14)$$

where δ is the corresponding discriminator for adversarial training. The distribution loss can also be designed similarly to the texture. They are enforced differently according to each attribute's characteristics for better results.

Summary. The parameters of the whole network including E_C, E_T, E_S, D_F, and Γ are optimized by minimizing the above three types of objectives, i.e., realism loss, reconstruction loss, and distribution loss. More details can be found in the supplementary material.

3.3 Discussion

Overall, CtrlHair has three differences with existing methods as shown in Table 1. Firstly, CtrlHair supports more interaction modes including reference photos, painted masks and sliding a set of bars. Secondly, CtrlHair can continuously and finely manipulate hair attributes such as hue of color, length of shape, while existing methods can only transfer the entire attribute. Thirdly, we elaberately consider the alignment between hair and face, and inpainting of those uncovered regions, while existing methods directly ignore or throw them to GAN.

4 Experiments

In this section, we present the experimental setting, then compare our CtrlHair method with existing methods in terms of hairstyle transfer, and investigate the ability of continuously and finely editing followed by an ablation study.

4.1 Datasets and Implementation Details

For experimental evaluation, the high-quality CelebAMask-HQ [13] dataset and FFHQ [11] dataset are used. From the two datasets, we select 10413 images without severe hat-occlusion and large pose orientation for experiments. Among them, 1000 are randomly sampled for testing and the rest 9413 images are used for training. The resolution of each image is resized to 256×256.

The encoder of color E_C, encoder of texture E_T and decoder for both of them D_X are all designed as fully connection architecture while shape encoder E_S and shape adaptor Γ are designed as convolution architecture considering that shape is position sensitive. The dimensionality of latent representations Z_C, Z_T, Z_S are set as 4, 8 and 16 respectively. For texture attribute, besides unsupervised mining, we also manually annotate a small number of images (1180 for wavy, and 727 for straightness) with a binary label for curliness factor to do weakly supervised training. In total, we edit 3 attributes with 28 controllable dimensions. Among them, 11 fine-grained dimensions are manually found to be with obvious semantics, which are shown in the supplementary materials.

4.2 Comparison with existing methods on hairstyle transfer

Since existing methods can not do continuous hair editing, we firstly compare with them on the task of discretely hair editing, i.e. hairstyle transfer. Our CtrlHair method is compared to MichiGAN [24], LOHO [21], and Barbershop [28], in terms of editing correctness, realism, and computational efficiency. Among them, MichiGAN needs an additional inpainting module for uncovered region, but this part is not publicly available, so we use GatedConv [26] for its inpainting.

Editing Correctness. The editing results of the compared methods are shown in Fig. 4, where all methods transfer the appearance of a reference photo and its corresponding shape to a given input image. Firstly, we compare the results of hair-appearance transferring as shown in Fig. 4(a), (b), and (c). As for the hair region, all methods transfer the hair appearance favorably, i.e. look very similar to the reference image. As for the appearance harmony of hair and face region, MichiGAN looks slightly stiff, LOHO, BarberShop and our CtrlHar look naturally. Secondly, we compare the results of hair-shape transferring. As can be seen in Fig. 4(d), when the reference's face shape is larger, it is inappropriate to directly replace the shape mask of the input image with the reference mask, e.g. artifact appears in MichiGAN and shadow appears on the left side of the face in LOHO, due to a lack of considering alignment. Both BarberShop and our CtrlHair achieve better results. In Fig. 4(e) and (f), MichiGAN and LOHO incorrectly inpaint the uncovered region, i.e. the uncovered region is inpainted

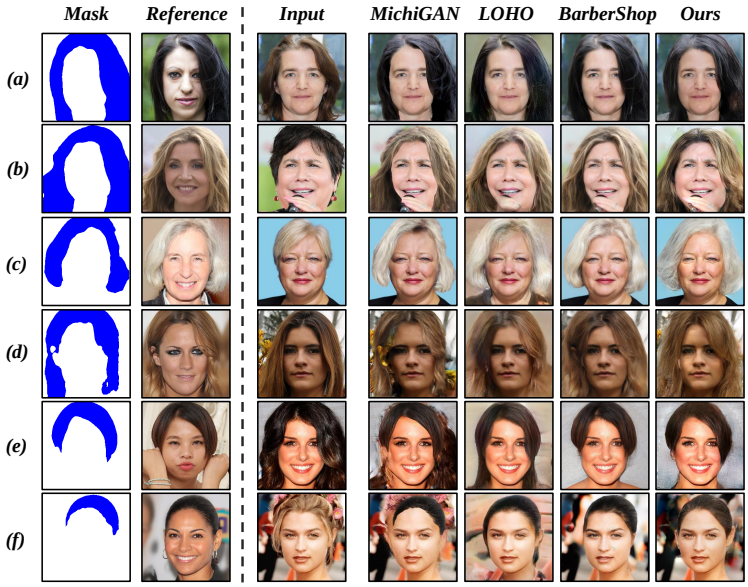


Fig. 4. Comparison with existing methods on hairstyle transfer with a reference image

Table 2. Comparison with the existing method in terms of realism and time efficiency

	MichiGAN [24]	LOHO [21]	BarberShop [28]	CtrlHair (ours)
FID↓	23.80	31.03	24.93	**21.39**
Time Consuming↓	**1.6s+3.1s**[1]	960.1s	395.8s	**7.7s**

with hair again while it should be inpainted with face or background. Our method produces more correct shape, benefiting from our elaborately designed shape adaptor. Please refer to the supplementary material for more comparison results.

Realism Comparison. To evaluate the generation realism, we compare the Fréchet Inception Distance (FID) [8] from different methods according to the same 3000 transferred images selected randomly. FID judges the similarity between edited images and real images. The results in Table 2 show that our method produces a lower FID than others, meaning a slightly better realism.

Time Consuming Comparison. The time consuming of inference on a single NVIDIA RTX is evaluated as shown in Table 2. Both LOHO and BarberShop need several minutes since they are optimization-based methods. As learning-based methods, both MichiGAN and our method only need several seconds, making them friendly for real-time interaction with users' manipulation.

[1] Since face parsing and inpainting processes are not publicly available for MichiGAN, BiSeNet [25] and GatedConv [26] are alternatively used, which cost 3.1s.

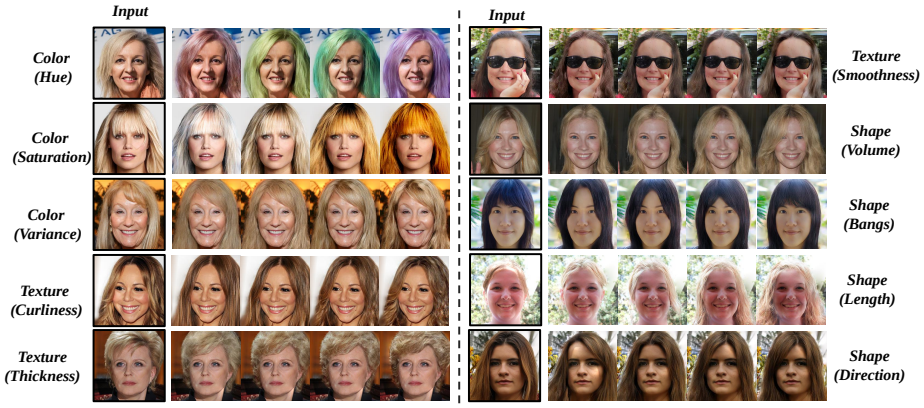


Fig. 5. Continuous hair editing at each dimension of an attribute via sliding a bar

Fig. 6. A wide variety of hair is obtained continuously for the person in the red box

4.3 Ability of Continuous Editing

The major contribution of our method is continuous and fine editing of hair. Fig. 1 shows some results of editing multiple attributes via sliding bars, and Fig. 5 further shows results of editing fine-grained variational factors of each attribute with no need of any reference image. These visual results show that our CtrlHair can correctly and separately edit the hair, i.e. correctly edit the desirable attribute while not affect other hair attributes or face region, in a continuous manner. Based on this capability, a wide variety of hair can be obtained continuously in Fig. 6, and continuous editing towards a target style indicated by a reference image can be naturally obtained as shown in Fig. 7.

Moreover, we quantitatively verify the ability of continuous editing of our method with results shown in Fig. 8. We take three attributes for examplar,

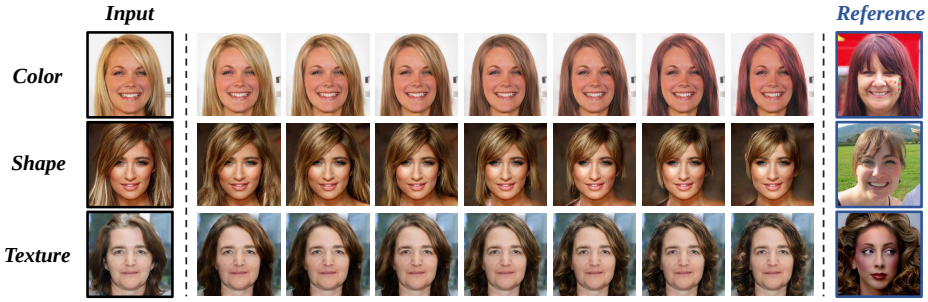
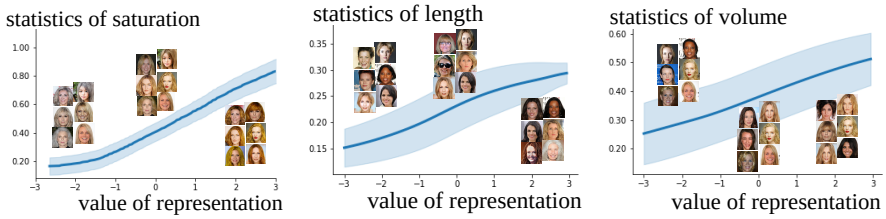


Fig. 7. Continuous editing towards a target style indicated by a reference image



(a) Saturation of Color (b) Length of Shape (c) Volume of Shape

Fig. 8. Illustration of continuous hair editing via quantitative curve for three attributes. Each curve shows attribute value of edited image w.r.t. the latent representation value

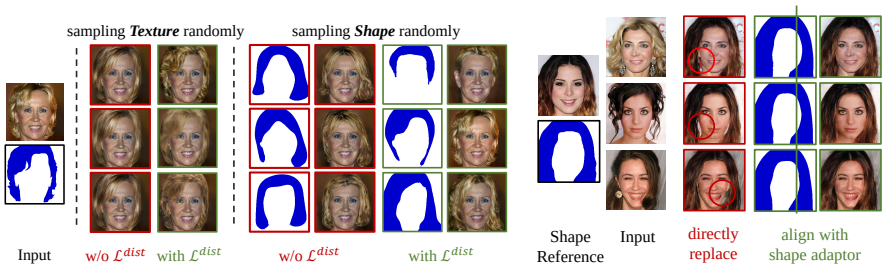
including saturation of color, volume of shape, and length of shape since the real value of these attributes are easily calculated as ground-truth for analysis. We continuously change the value of the latent representation (i.e. horizontal axis), such as saturation of color. Then, portrait images are generated with these latent representations. After that, the values corresponding to the manipulated factor in the generated image are calculated as values in the vertical axis, such as the real saturation of hair region of generated image. Each curve in Fig. 8 is plotted according to 100 sampling points during the interval $[-3\sigma, 3\sigma]$. Each point is calculated as an average 900 images with standard variance shown as blue region around the curve. As can be seen, the latent representation affects the attributes continuously and smoothly, validating the ability of our CtrlHair.

Please refer to the supplementary material for more results, more description of the values in Fig. 8, and editing demos with a user interface.

4.4 Ablation Study

In this section, we verify the effectiveness of standard multivariate Gaussian distribution, and necessity of shape adaptor.

Effectiveness of Gaussian Distribution. To evaluate how the standard multivariate Gaussian distribution affects the control of editing, we compare



(a) with or w/o Gaussian distribution constraints (b) with or w/o shape adaptor

Fig. 9. Ablation Study of the constraints of standard multivariate Gaussian distribution and shape adaptor.

the hair editing results with and without it by randomly sampling texture and shape latent representations from Gaussian distribution. The results are shown in Fig. 9(a). It can be seen that, without Gaussian distribution, changing the latent representation can only slightly change the texture and shape since the hairs attributes are mapped to an expansive space, making them hardly be sampled. When with standard multivariate Gaussian distribution, changing the latent representation can traverse a reasonable range with rich variations.

Necessity of Shape Adaptor. We investigate our CtrlHair method with and without shape adaptor, i.e. directly replace the hair mask of an input image with that of the reference image. The results are shown in Fig. 9(b). It can be clearly seen that, in the edited images without using the shape adaptor, a large-shaped hair and a thin face does not match well, leading to awkward results. This comparison demonstrates the necessity and effectiveness of our shape adaptor.

5 Conclusion and Future Work

In this work, we propose an efficiently controllable editing method for hair, which can provide a set of bars for users to simply slide it to do continuous and fine hair editing. The proposed method also naturally supports editing with reference photos and user-painted mask. The method is designed as an encoding-editing-decoding framework, where the latent representations for each attribute of hair are formulated as standard multivariate Gaussian distribution supporting for continuous and fine editing. In this work, most variational factors of attributes are mined unsupervised, which does not necessarily correspond to a meaning semantic. In the future, we will explore disentangling them according to the intrinsic semantic of hair attributes.

Acknowledgement. This work is partially supported by the National Key Research and Development Program of China (No. 2017YFA0700800), the Natural Science Foundation of China (No. 62122074), and the Beijing Nova Program (Z191100001119123).

References

1. Burgess, C.P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., Lerchner, A.: Monet: Unsupervised scene decomposition and representation. arXiv preprint arXiv:1901.11390 (2019)
2. Chai, M., Ren, J., Tulyakov, S.: Neural hair rendering. In: European Conference on Computer Vision (ECCV) (2020)
3. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Neural Information Processing Systems (NIPS) (2016)
4. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
5. Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Neural Information Processing Systems (NIPS) (2014)
7. He, Z., Zuo, W., Kan, M., Shan, S., Chen, X.: Attgan: Facial attribute editing by only changing what you want. IEEE Transactions on Image Processing (T-IP) **28**(11), 5464–5478 (2019)
8. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Neural Information Processing Systems (NIPS) (2017)
9. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision (ECCV) (2016)
10. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: International Conference on Learning Representations (ICLR) (2018)
11. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
12. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
13. Lee, C.H., Liu, Z., Wu, L., Luo, P.: Maskgan: towards diverse and interactive facial image manipulation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
14. Liu, M., Ding, Y., Xia, M., Liu, X., Ding, E., Zuo, W., Wen, S.: Stgan: A unified selective transfer network for arbitrary image attribute editing. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
15. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
16. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: International Conference on Machine Learning (ICML) (2017)
17. Or-El, R., Sengupta, S., Fried, O., Shechtman, E., Kemelmacher-Shlizerman, I.: Lifespan age transformation synthesis. In: European Conference on Computer Vision (ECCV) (2020)

18. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
19. Perarnau, G., Van De Weijer, J., Raducanu, B., Álvarez, J.M.: Invertible conditional gans for image editing. arXiv preprint arXiv:1611.06355 (2016)
20. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. International Conference on Learning Representations (ICLR) (2015)
21. Saha, R., Duke, B., Shkurti, F., Taylor, G.W., Aarabi, P.: Loho: Latent optimization of hairstyles via orthogonalization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
22. Shen, W., Liu, R.: Learning residual images for face attribute manipulation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
23. Shoshan, A., Bhonker, N., Kviatkovsky, I., Medioni, G.: Gan-control: Explicitly controllable gans. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
24. Tan, Z., Chai, M., Chen, D., Liao, J., Chu, Q., Yuan, L., Tulyakov, S., Yu, N.: Michigan: Multi-input-conditioned hair image generation for portrait editing. ACM Transactions on Graphics (TOG) **39**(4), 95 (2020)
25. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: European Conference on Computer Vision (ECCV) (2018)
26. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
27. Zhang, G., Kan, M., Shan, S., Chen, X.: Generative adversarial network with spatial attention for face attribute editing. In: European Conference on Computer Vision (ECCV) (2018)
28. Zhu, P., Abdal, R., Femiani, J., Wonka, P.: Barbershop: Gan-based image compositing using segmentation masks. ACM Transactions on Graphics (TOG) **40**(6), 215:1–215:13 (2021)
29. Zhu, P., Abdal, R., Qin, Y., Wonka, P.: Sean: Image synthesis with semantic region-adaptive normalization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
30. zllrunning: face-parsing.pytorch. `https://github.com/zllrunning/face-parsing.PyTorch`

GAN with Multivariate Disentangling for Controllable Hair Editing - Supplementary Material

Xuyang Guo[1,2], Meina Kan[1,2], Tianle Chen[1,2], and Shiguang Shan[1,2,3]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
{xuyang.guo,tianle.chen}@vipl.ict.ac.cn, {kanmeina,sgshan}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beiijng 100049, China
[3] Peng Cheng Laboratory, Shenzhen, 518055, China

This supplementary material provides more details in six aspects: Section A gives more details about network architectures and the training procedure. Section B gives more details about how to automatically get color annotations. Section C presents details about exploring binary (i.e. weak) annotation to guide the semantic dimension disentangling for texture. Section D discusses the comparison between our method and other general GAN inversion-based editing methods. Section E shows details about testing of continuous editing ability. Section F shows more results about the disentangled dimension of color, texture, and shape, more comparisons with existing methods, and more visual results.

A Network Architectures and Training Procedure

Network Architectures for Hair Color and Texture Editing. The networks of color and texture have two parts, encoders (E_C, E_T) and decoder (D_X).

The encoder E_C of color is designed as a fully connection architecture with 3 hidden layers, as shown in Fig. 1. It takes hair feature X as input to get the latent representations Z_C. The encoder E_T of texture is designed as a fully connection architecture with 4 hidden layers, as shown in Fig. 2. It takes hair feature X as input to get the latent representations Z_T. A single decoder D_X is used to generate the edited hair feature $\hat{X}$ from edited latent representations $\hat{Z}_C, \hat{Z}_T$ of color and texture. The network architecture of the decoder D_X has 4 hidden layers, as shown in Fig. 3.

During optimization of the encoder and decoder, adversarial training with $\mathcal{L}^{real}$ is needed to ensure the reality of edited feature $\hat{X}$. So, a discriminator δ_X is introduced as described in Section 3.2 of our paper. The discriminator has a similar architecture with E_T and shares network weights as shown in Fig. 2 for saving parameters and better extracting common features.

Network Architecture for Hair Shape Editing. The networks for the shape editing are designed as convolution architecture considering that shape is position sensitive, consisting of the shape encoder and the shape adaptor. Since the input segmentation mask has no position information for convolution operation,

Algorithm 1: Training of CtrlHair

input : Images set $\Omega_I = \{I\}$ with its hair appearance features $\Omega_X = \{X\}$, color label $\Omega_{\tilde{Z}_C} = \{\tilde{Z}_C\}$ and portrait masks $\Omega_S = \{S\}$;
output: Encoders E_C, E_T, E_S, decoders D_X and shape adaptor Γ;
Initialize $E_C, E_T, E_S, D_X, \Gamma$ and discriminators $\delta_X, \delta_S, \delta_{Z_S}$;
while *not converge* **do**
`// training encoders and decoder for color and texture`
Sample a batch of latent representations $\hat{Z}_C, \hat{Z}_T \in \mathcal{N}(0, I)$;
Generate $\hat{X}$ using Eq. (4);
Sample a batch of appearance features $X \in \Omega_X$ with its color labels $\tilde{Z}_C \in \Omega_{\tilde{Z}_C}$;
Update E_C according to $\mathcal{L}_C^{dist}$, E_T according to $\mathcal{L}^{real}, \mathcal{L}_T^{rec}, \mathcal{L}_T^{dist}$, and D_X according to $\mathcal{L}^{real}, \mathcal{L}_C^{rec}, \mathcal{L}_T^{rec}, \mathcal{L}_T^{dist}$;
`// training encoder and adaptor for shape`
Sample a batch of masks from Ω_S and extract their hair masks S_H;
Sample another batch of masks from Ω_S and extract their face and background masks (S_F, S_B);
Generate adjusted portrait mask $\hat{S}_k$ from S_H, S_F, S_B using Eq. (2) and (5);
Update E_S according to $\mathcal{L}^{real}, \mathcal{L}_S^{rec}, \mathcal{L}_S^{dist}$, and Γ according to $\mathcal{L}^{real}, \mathcal{L}_S^{rec}$;
`// training discriminators`
Sample a batch of true appearance feature $X^r \in \Omega_X$, portrait mask $S^r \in \Omega_S$ and true latent representations $Z_S^r \in \mathcal{N}(0, I)$;
Update $\delta_X, \delta_S, \delta_{Z_S}$ according to their respective adversarial training losses;
end

inspired by NeRF [6], $1^{st} \sim 10^{th}$ order sine and cosine position embeddings are additionally concatenated for each input segmentation mask.

The shape encoder E_S takes the above-mentioned input to obtain the shape latent representation Z_S, and the encoder's network architecture is shown in Fig. 4. Then, the shape adaptor Γ takes latent representation $\hat{Z}_S$ of edited shape, face mask S_F, and background mask S_B as input, to generate edited portrait mask $\hat{S}$. The shape adaptor Γ's network architecture is as shown in Fig. 7.

During training, adversarial training is employed to ensure the reality of edited portrait mask $\hat{S}$ by $\mathcal{L}^{real}$ and the standard multivariate gaussian distribution of encoded shape latent representation Z_S by $\mathcal{L}_S^{dist}$. So two discriminators δ_S and δ_{Z_S} are designed. The network architectures are shown in Fig. 5 and 6.

Training Details. We use Adam optimizer [5]. The batch size is set as 16. For loss weights, we set $\lambda^{real} = 1$, $\lambda_C^{rec} = \lambda_C^{dist} = 10^{-2}$, $\lambda_T^{rec} = 10^2$, $\lambda_T^{dist} = 10^{-2}$, $\lambda_S^{rec} = 10^2$, $\lambda_S^{dist} = 1$. For all encoders, decoders, and the shape adaptor, the learning rate is set as 2×10^{-4}. For all discriminators, the learning rate is set as 1×10^{-4}. The training procedure is shown in Algorithm 1. To achieve better results, the shape branch can also be trained separately, i.e., separating the steps related to shape editing for better controlling the network parameters.

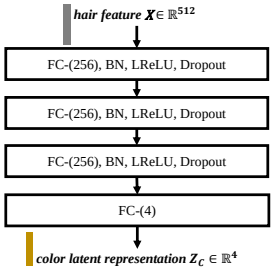


Fig. 1. Color encoder E_C

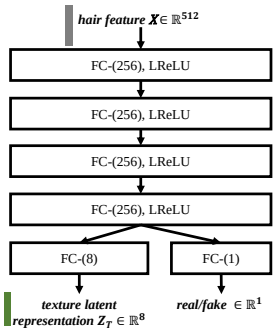
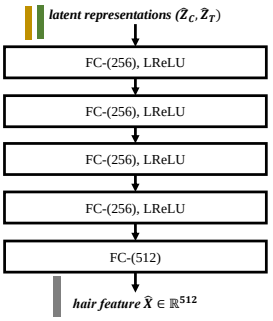


Fig. 2. The shared texture encoder E_T and the feature discriminator δ_X



Network architecture of the feature decoder D_X

Fig. 3. Feature decoder D_X for color and texture

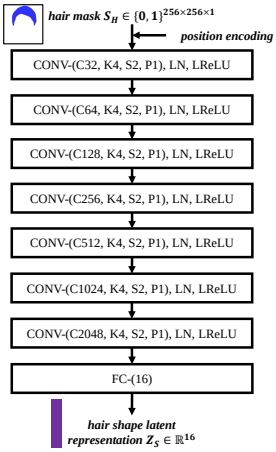


Fig. 4. Shape encoder E_S

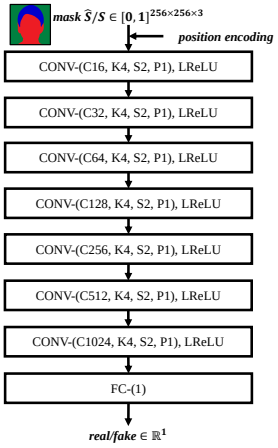


Fig. 5. Mask discriminator δ_S

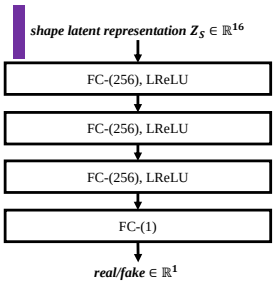


Fig. 6. Discriminator δ_{Z_S} of shape latent representation

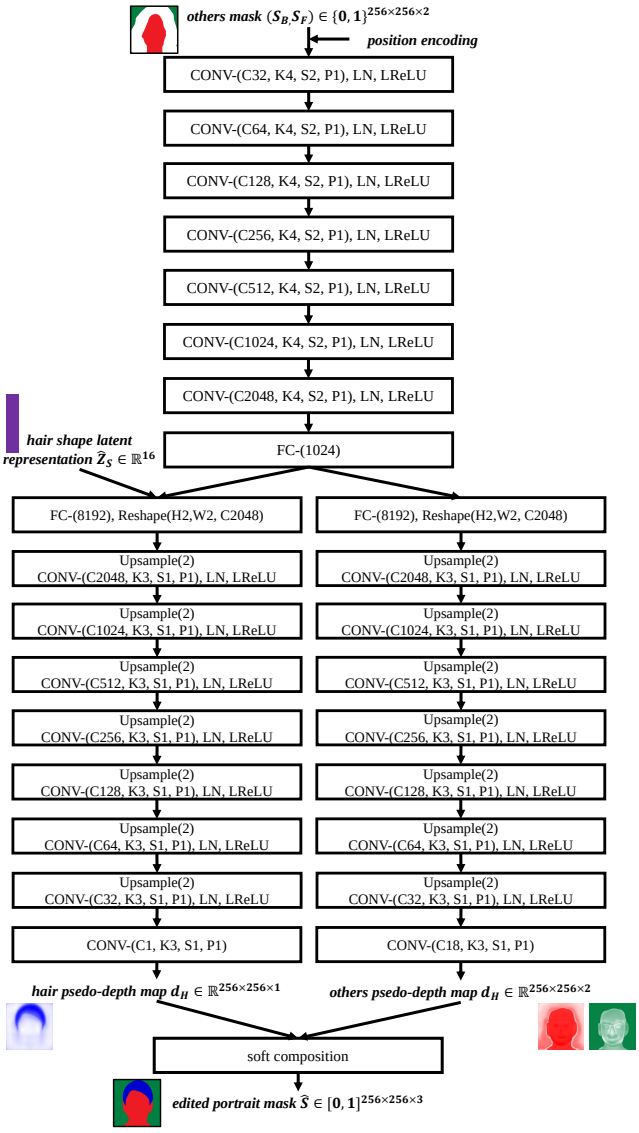


Fig. 7. Shape adaptor Γ

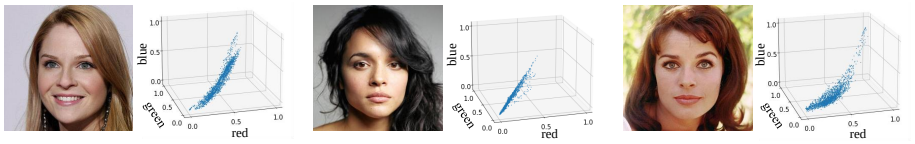


Fig. 8. The scatter plot of hair pixels for each image

B Details about Automatic Hair Color Annotation

In Section 3.2 of our paper, we briefly describe that the latent representation of hair color is designed as a 4-dimensional vector Z_C, indicating the major HSV values and color variance. They are automatically calculated as below, with no need for manual labeling.

For *HSV color values*, as denoted in our paper, the mean of pixels in the hair region is used. Besides mean, we also try median and mode, but their effects are slightly worse than the mean. For *color variance*, we observe that for most images, all pixels of hair region are roughly distributed on a one-dimensional manifold in RGB space for a single portrait as shown in Fig. 8. Based on this observation, for each image, we employ PCA to calculate its first principal component (containing about 97.0% energy of hair color pixels on average) and take the variance corresponding to the first principal component as the color variance.

C Details about Hair Texture Editing guided by Binary Annotation

As denoted in Section 3.2 of our paper, unsupervised training of hair texture disentangling and editing is able to achieve continuous and fine editing of different dimensions of the texture. However, each dimension of the learned latent representation Z_T does not necessarily correspond to an explicit semantic meaning. This is because the texture hardly obtains continuous label for supervised training, which is quite different from the color that is easy to obtain a continuous label.

However, sometimes binary annotation (which is discrete and weak) can be obtained, such as wavy or straight. With this kind of weak label, our method is also easily compatible for better editing by introducing an additional binary classification loss like that in [4], detailed in the following.

Taking the curliness factor as an example, a small number of samples are binarily labeled, including 1180 images labeled as wavy and 727 images labeled as straight. Specifically, one dimension in the texture latent representation Z_T is used to represent curliness, denoted as Z_T^{cur}, and the rest 7 dimensions of Z_T remain unchanged, i.e. still in an unsupervised manner. In addition to adopting the constraints indicated in Eq. (9), (10), (11) of our paper for texture, additionally, a binary classification loss is introduced for curliness.

Firstly, a binary label $\hat{Y}$ is assigned for the edited latent representation $\hat{Z}_T^{cur}$, i.e. $\hat{Y} = \mathbb{I}(\hat{Z}_T^{cur} \geqslant 0)$. Then, the decoded feature $\hat{X}$ is input to a pre-trained binary classifier P to get its prediction. After that, the classification loss between the estimated prediction and the binary label is calculated as follows:

$$\min_{D_X} \mathcal{L}_{T,cur}^{rec} = \mathbb{E}_{\hat{Z}_C, \hat{Z}_T \sim \mathcal{N}(0,I)} \left[-\hat{Y} \log P(\hat{X}) \right], \tag{1}$$

where P is the pre-trained binary classifier for curliness Y from the hair feature X trained by using truly labeled data. This loss can enforce that the change of latent representation is consistent with the binary label, which roughly guides the learning of continuous editing ability for curliness.

D Comparison with GAN Inversion-based Editing Methods

General GAN inversion-based editing methods, such as [1,2,3,7], also have the ability of continuous editing of portrait images and show compelling results. However, these methods are not quite the same as ours in terms of tasks. Usually for hair editing, only those editing that can manipulate the semantic variation of hair such as hair shape, structure, etc., are considered meaningful. The general GAN inversion-based editing methods can only achieve continuous editing, but not meaningful editing. BarberShop [10] and LOHO [8], also as GAN inversion-based methods, are designed specifically for hair, which is meaningful but loses continuity as discussed in the text of our paper. In one word, on premising meaningful editing, our method is continuous while existing methods are not.

E Details about Testing of Continuous Editing Ability

In section 4.3 and Fig. 8 of our paper, we test the ability of continuous editing, specifically, taking saturation of color, length of shape, and volume of shape as three examples. In testing, rough true values of the three attributes are used as criteria for evaluation. Here we introduce how these values are calculated in detail.

Saturation of Color: The mean of the saturation of all pixels in the hair region is directly used to represent the saturation of hair color.

Length of Shape: The length is mainly reflected in the vertical distance of the hair shape. In order to estimate the length and avoid being too sensitive to outliers, we use the standard deviation of the vertical axis coordinates of all pixels in hair region to reflect the length of hair, i.e. the larger the deviation is, the longer the hair is.

Volume of Shape: The hair volume reflects the amount of hair, which is reflected by the ratio of hair area and the whole image to roughly approximate hair volume.

The calculation of the three criteria is not exactly the same as the ground truth, but it is certainly proportional to the ground truth. So the relationship

Table 1. Summary of attributes and fine-grained factors of our method

Attributes	Color	Texture	Shape
Dimensions	4	8	16
Fine-grained Factors	Hue Saturation Brightness Variance	Curliness Smoothness Hair-strand-thickness	Length Volume Bangs Direction

between the above criteria and the corresponding latent representation in Fig. 8 of our paper can show whether continuous editing ability is correctly achieved.

F More Visual Results

In total, our method can edit 3 attributes (i.e. color, texture, and shape) of hair, with 28 controllable dimensions. Among them, 11 dimensions are observed to be with obvious semantics as shown in Table 1. Besides, more visual results are shown for a better understanding of our method.

Fig. 9 shows a group of editing examples by using CtrlHair's Demo. These results illustrate that our method is convenient and friendly for user interaction by sliding bars.

Fig. 10 shows more comparison results with existing methods on hairstyle transfer with a reference image. From the figure, the effect of our method on color and texture is comparable to other methods. While our method is obviously superior to MichiGAN [9] and LOHO [8] in shape transferring.

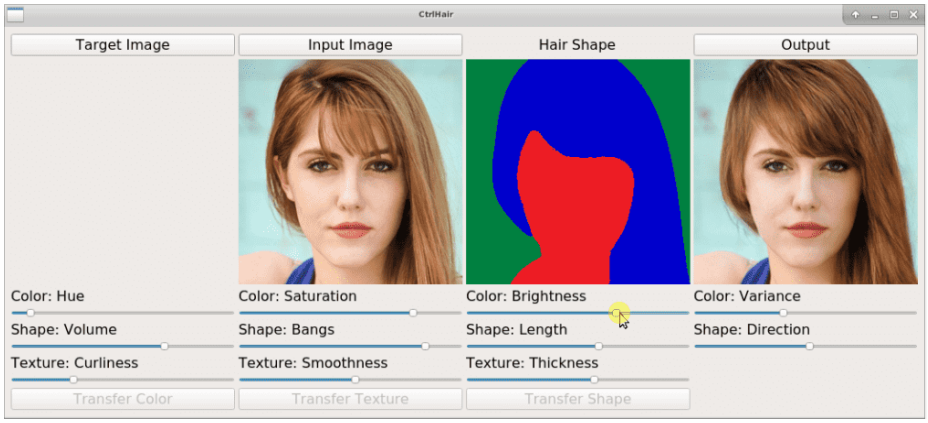
Fig. 11 shows more editing of fine factors of all attributes simultaneously and separately from our CtrlHair. Fig. 12∼22 show more editing results of each fine factor of attributes. These figures illustrate the ability of fine and continuous editing of our method.

Finally, Fig. 23 shows the result of hair editing with a wide variety for a given person, which illustrates the arbitrariness of style achieved by our method.

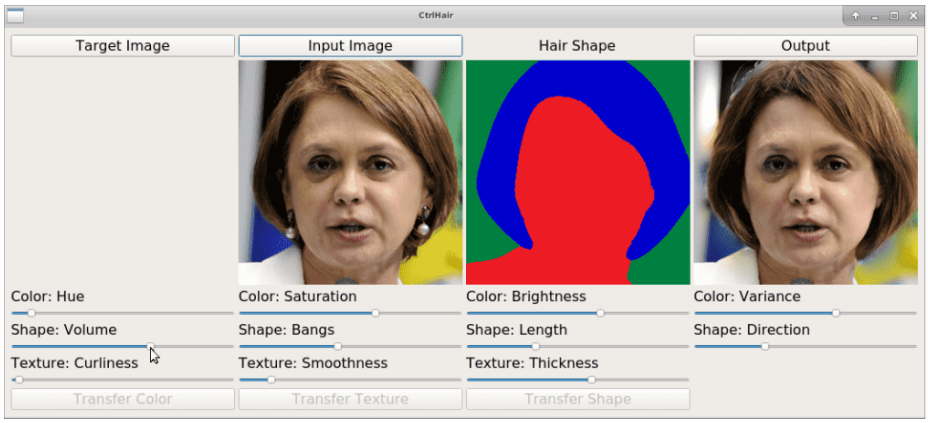
References

1. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
2. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
3. Alaluf, Y., Patashnik, O., Cohen-Or, D.: Restyle: A residual-based stylegan encoder via iterative refinement. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
4. Kaneko, T., Hiramatsu, K., Kashino, K.: Generative adversarial image synthesis with decision tree latent controller. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

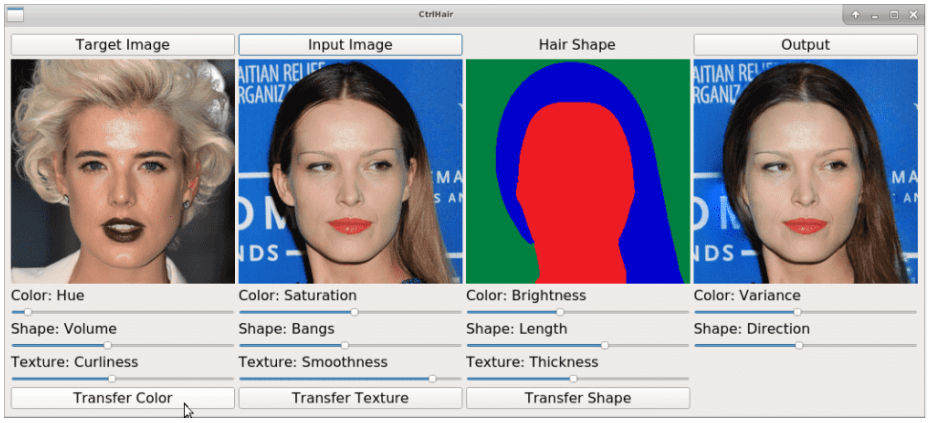
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision (ECCV) (2020)
7. Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., Cohen-Or, D.: Encoding in style: a stylegan encoder for image-to-image translation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
8. Saha, R., Duke, B., Shkurti, F., Taylor, G.W., Aarabi, P.: Loho: Latent optimization of hairstyles via orthogonalization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
9. Tan, Z., Chai, M., Chen, D., Liao, J., Chu, Q., Yuan, L., Tulyakov, S., Yu, N.: Michigan: Multi-input-conditioned hair image generation for portrait editing. ACM Transactions on Graphics (TOG) **39**(4), 95 (2020)
10. Zhu, P., Abdal, R., Femiani, J., Wonka, P.: Barbershop: Gan-based image compositing using segmentation masks. ACM Transactions on Graphics (TOG) **40**(6), 215:1–215:13 (2021)



(a) Editing color



(b) Editing shape and texture



(c) Hybrid Editing of transfering and fine-tuning with sliding bars

Fig. 9. A group of editing examples using CtrlHair's Demo with User Interface. Please click on the images to open the animation

Fig. 10. Comparison with existing methods on hairstyle transfer with a reference image

Fig. 11. Editing fine factors of attributes simultaneously and separately. For a given portrait in the upper left corner of each subfigure, CtrlHair can edit the hair by sliding a set of bars

Fig. 12. Continuous editing of hue of color

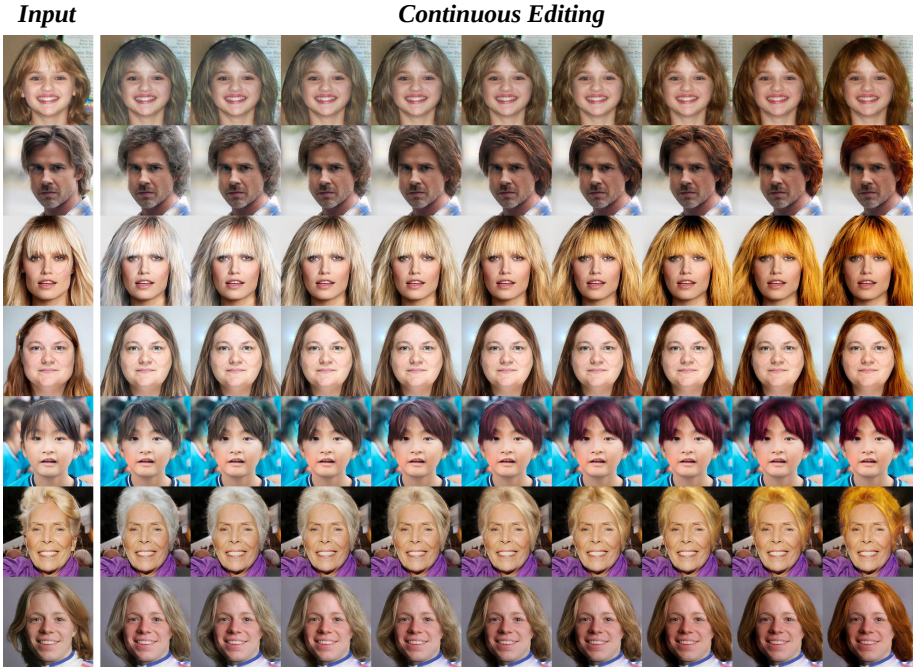


Fig. 13. Continuous editing of saturation of color

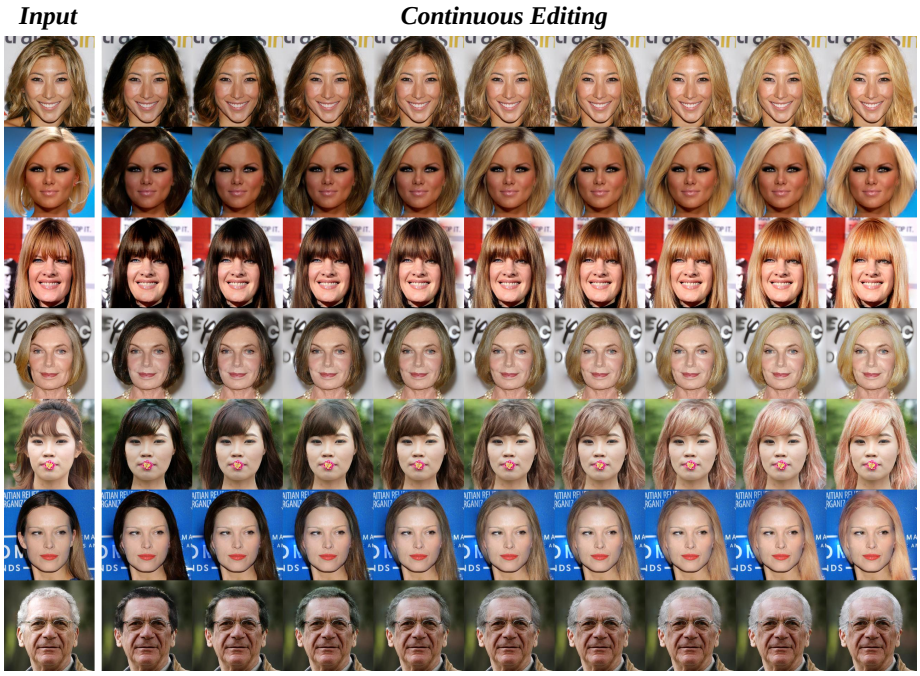


Fig. 14. Continuous editing of brightness of color

Fig. 15. Continuous editing of variance of color

Fig. 16. Continuous editing of curliness of texture

Fig. 17. Continuous editing of smoothness of texture

Fig. 18. Continuous editing of hair-strand-thickness of texture

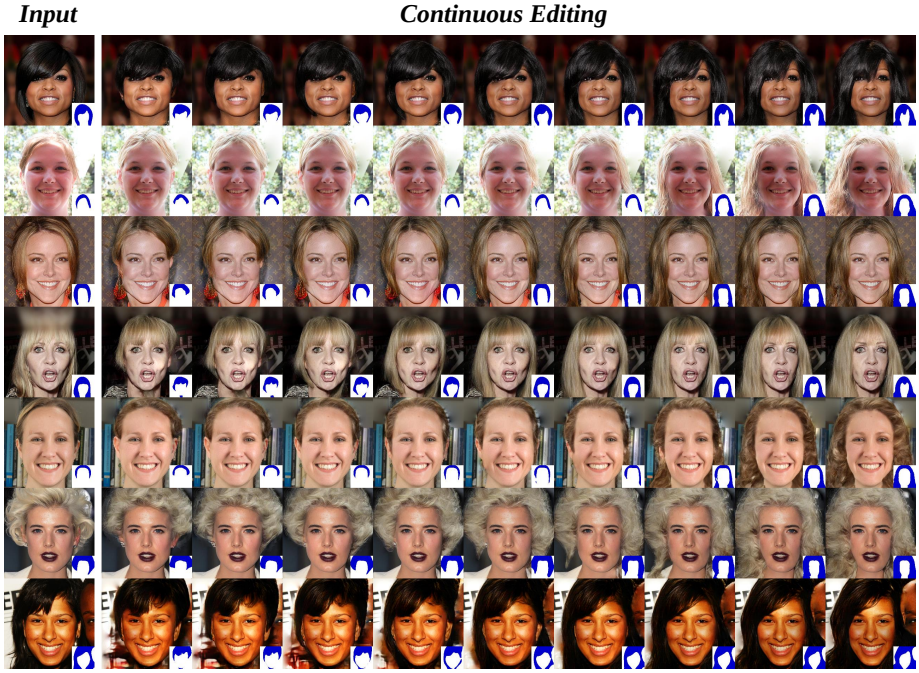


Fig. 19. Continuous editing of length of shape

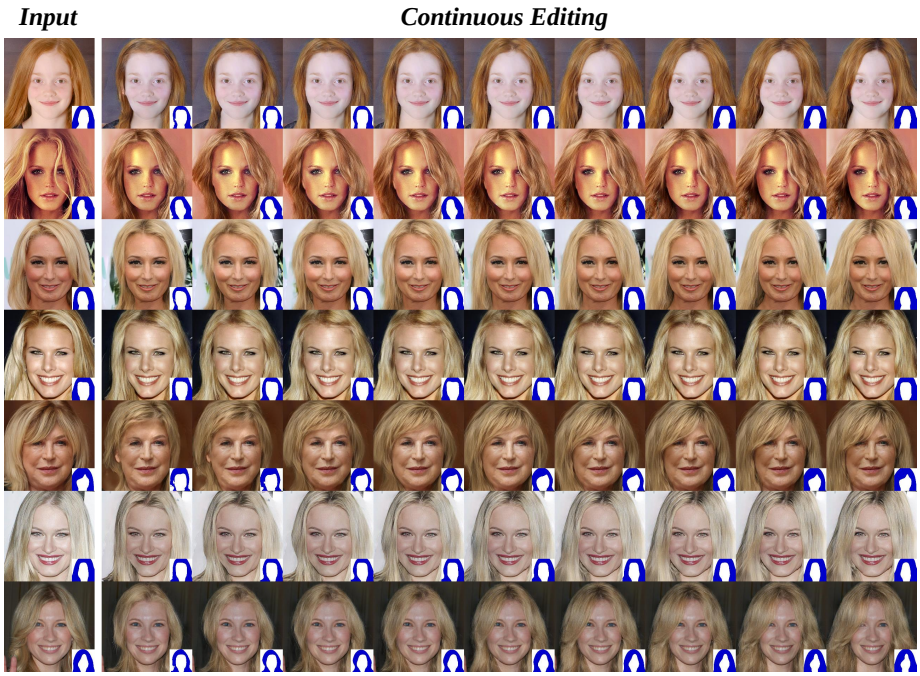


Fig. 20. Continuous editing of volume of shape

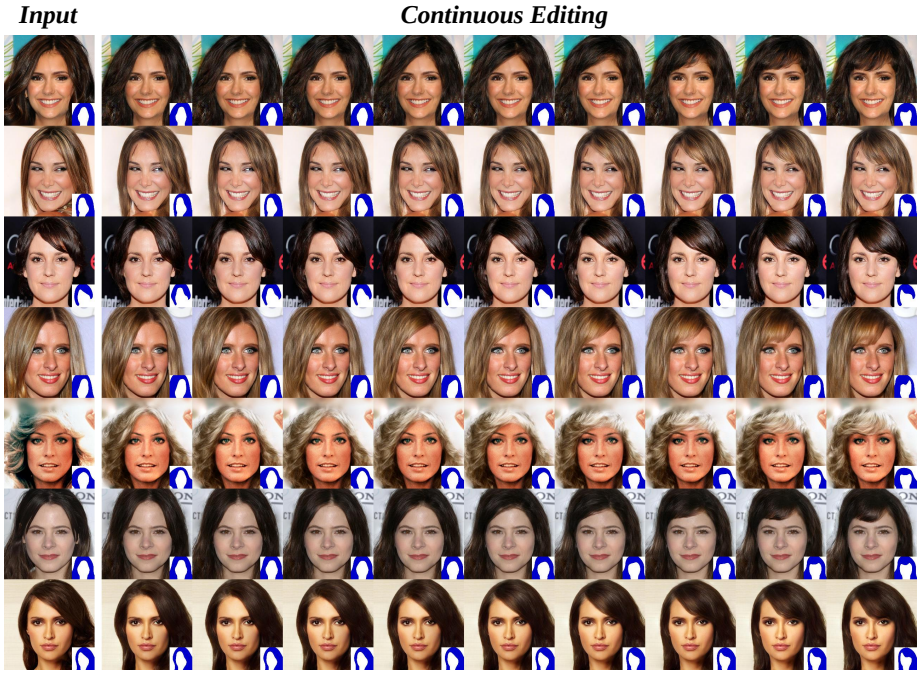


Fig. 21. Continuous editing of bangs of shape

Fig. 22. Continuous editing of direction of shape

Fig. 23. A wide variety of hair is obtained for the person in the red box