Face-to-Parameter Translation for Game Character Auto-Creation

Tianyang Shi
NetEase Fuxi AI Lab
shitianyang@corp.netease.com

Yi Yuan*
NetEase Fuxi AI Lab
yuanyi@corp.netease.com

Changjie Fan
NetEase Fuxi AI Lab
fanchangjie@corp.netease.com

Zhengxia Zou
University of Michigan, Ann Arbor
zzhengxi@umich.edu

Zhenwei Shi*
Beihang University
shizhenwei@buaa.edu.cn

Yong Liu
Zhejiang University
yongliu@iipc.zju.edu.cn

Abstract

Character customization system is an important component in Role-Playing Games (RPGs), where players are allowed to edit the facial appearance of their in-game characters with their own preferences rather than using default templates. This paper proposes a method for automatically creating in-game characters of players according to an input face photo. We formulate the above "artistic creation" process under a facial similarity measurement and parameter searching paradigm by solving an optimization problem over a large set of physically meaningful facial parameters. To effectively minimize the distance between the created face and the real one, two loss functions, i.e. a "discriminative loss" and a "facial content loss", are specifically designed. As the rendering process of a game engine is not differentiable, a generative network is further introduced as an "imitator" to imitate the physical behavior of the game engine so that the proposed method can be implemented under a neural style transfer framework and the parameters can be optimized by gradient descent. Experimental results demonstrate that our method achieves a high degree of generation similarity between the input face photo and the created in-game character in terms of both global appearance and local details. Our method has been deployed in a new game last year and has now been used by players over 1 million times.

1. Introduction

The character customization system is an important component in role-playing games (RPGs), where players are allowed to edit the profiles of their in-game characters according to their own preferences (e.g. a pop star or even themselves) rather than using default templates. In recent RPGs, to improve the player's immersion, character customization systems are becoming more and more sophisticated. As a result, the character customization process turns out to be time-consuming and laborious for most players. For example, in "Grand Theft Auto Online[1]", "Dark Souls III[2]", and "Justice[3]", to create an in-game character with a desired facial appearance according to a real face photo, a player has to spend several hours manually adjusting hundreds of parameters, even after considerable practice.

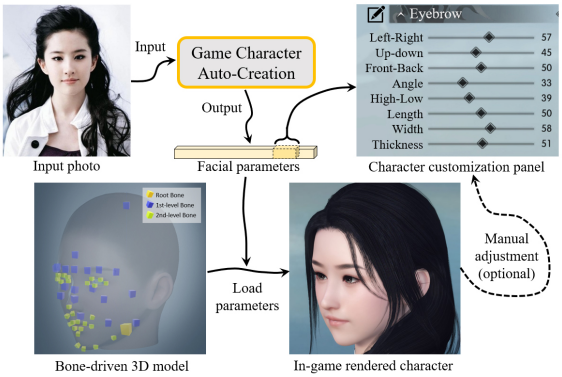

Figure 1. An overview of our method: We propose a method for game character auto-creation based on an input face photo, which can be formulated under a facial similarity measurement and the searching of a large set of physically meaningful facial parameters. Users can further optionally fine-tune the facial parameters on our creation according to their needs.

A standard work-flow for the creation of the character's face in RPGs begins with the configuration of a large set of facial parameters. A game engine then takes in these user-specified parameters as inputs and generates the 3D faces. Arguably, the game character customization can be considered as a special case of a "monocular 3D face reconstruc-

*Corresponding authors

[1]https://www.rockstargames.com/GTAOnline
[2]https://www.darksouls.jp
[3]https://n.163.com

tion" [2, 34, 37] or a "style transfer" [10–12] problem. Generating the semantic content and 3D structures of an image have long been difficult tasks in the computer vision field. In recent years, thanks to the development of deep learning techniques, computers are now able to automatically produce images with new styles [12, 16, 25] and even generate 3D structures from single facial image [34, 37, 39] by taking advantages of deep Convolutional Neural Networks (CNN).

But unfortunately, the above methods cannot be directly applied in the game environment. The reasons are threefold. First, these methods are not designed to generate parameterized characters, which is essential for most game engines as they usually take in the customized parameters of a game character rather than images or 3D meshgrids. Second, these methods are not friendly to user interactions as it is extremely difficult for most users to directly edit the 3D meshgrids or rasterized images. Finally, the rendering process of a game engine given a set of user-specified parameters is *not differentiable*, which has further restricted the applicability of deep learning methods in the game environment.

Considering the above problems, this paper proposes a method for the automatic creation of an in-game character according to a player's input face photo, as shown in Fig. 1. We formulate the above "artistic creation" process under a facial similarity measurement and a parameter searching paradigm by solving an optimization problem over a large set of facial parameters. Different from previous 3D face reconstruction approaches [2, 34, 37] that produce 3D face meshgrids, our method creates 3D profile for a bone-driven model by predicting a set of facial parameters with a clear physical significance. In our method, each of the parameters controls an individual attribute of each facial components, including the position, orientation and scale. More importantly, our method supports additional user interactions on the basis of the creating results, where players are allowed to make further improvements on their profiles according to their needs. As the rendering process of a game engine is not differentiable, a generative network G is designed as an "imitator" to imitate the physical behavior of the game engine, so that the proposed method can be implemented under a neural style transfer framework and the facial parameters can be optimized by using gradient descent, thus we refer to our method as a "Face-to-Parameter (F2P)" translation method.

As the facial parameter searching in our method is essentially a cross-domain image similarity measurement problem, we take advantages of the deep CNN and multi-task learning to specifically design two kinds of loss functions, i.e. a "discriminative loss" and a "facial content loss" – the former corresponding to the similarity measurement of the global facial appearance and the latter focusing more on local details. Due to the all CNN design, our model can be optimized in a unified end-to-end framework. In this way, the input photo can be effectively converted to a realistic in-game character by minimizing the distance between the created face and the real one. Our method has been deployed in a new game since Oct. 2018 and now has been providing over 1 million times services.

Our contributions are summarized as follows:

1) We propose an end-to-end approach for face-to-parameter translation and game character auto-creation. To our best knowledge, there are few previous works have been done on this topic.

2) As the rendering process of a game engine is not differentiable, we introduce an imitator by constructing a deep generative network to imitate the behavior of a game engine. In this way, the gradient can smoothly back-propagate to the input so that the facial parameters can be updated by gradient descent.

3) Two loss functions are specifically designed for the cross-domain facial similarity measurement. The proposed objective can be jointly optimized in a multi-task learning framework.

2. Related work

Neural style transfer: The style transfer from one image onto another has long been a challenging task in image processing [10, 11]. In recent years, Neural Style Transfer (NST) has made huge breakthroughs in style transfer tasks [10–12], where the integration of deep convolutional features makes it possible to explicitly separate the "content" and "style" from input images. Most of the recent NST models are designed to minimize the following objective:

$$\mathcal{L}_{total} = \mathcal{L}_{content} + \lambda \mathcal{L}_{style}, \tag{1}$$

where $\mathcal{L}_{content}$ and $\mathcal{L}_{style}$ correspond to the constraints on the image content and the image style. λ controls the balance of the above two objectives.

Current NST methods can be divided into two groups: the global methods [5, 9, 12, 19, 21, 26, 29, 35, 36] and the local methods [6, 25, 27], where the former measures the style similarity based on the global feature statistics, while the latter performs the patch-level matching to better preserve the local details. To integrate both advantages of global and local methods, the hybrid method [16] is proposed more recently. However, these methods are specifically designed for image-to-image translation rather than a bone-driven 3D face model thus cannot be applied to in-game environments.

Monocular 3D face reconstruction: Monocular 3D face reconstruction aims to recover the 3D structure of the human face from a single 2D facial image. The traditional approaches of this group are the 3D morphable model (3DMM) [2] and its variants [3, 32], where a 3D face model

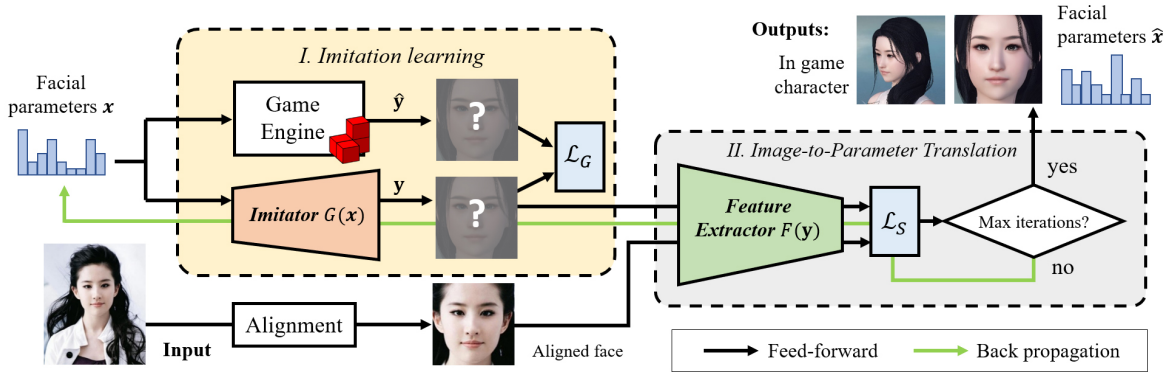

Figure 2. The processing pipeline of the proposed method. Our model consists of an Imitator $G(\boldsymbol{x})$ and and a Feature Extractor $F(\boldsymbol{y})$. The former aims to simulate the behavior of a game engine by taking in the user-customized facial parameters $\boldsymbol{x}$ and producing a "rendered" facial image $\boldsymbol{y}$. The latter determines the feature space in which the facial similarity measurement can be performed to search for an optimal set of facial parameters.

is first parameterized [14] and is then optimized to match a 2D facial image. In recent years, deep learning based face reconstruction methods are now able to achieve end-to-end reconstruction from a 2D image to 3D meshgrids [8, 34, 37–40]. However, these methods are not friendly for user interactions since it is not easy to edit on 3d meshgrids, and their generated face parameters lack explicit physical meanings. A similar work to ours is Genova's "differentiable renderer" [13], in which they directly render parameterized 3D face model by employing a differentiable rasterizer. In this paper, we introduce a more unified solution to differentiate and imitate a game engine regardless of the type of its renderer and 3D model structure by using a CNN model.

Generative Adversarial Network (GAN): In addition to the above approaches, GAN [15] has made great progress in image generation [1, 30, 31, 33], and has shown great potential in image style transfer tasks [4, 20, 28, 44, 46]. A similar approach with our method is Tied Output Synthesis (TOS) [41], which takes advantage of adversarial training to create parameterized avatars based on human photos. However, this method is designed to predict discrete attributes rather than continuous facial parameters. In addition, in RPGs, learning to directly predict a large set of 3D facial parameters from a 2D photo will lead to the defect of parameter ambiguity since the intrinsic correspondence from 2D to 3D is an "one-to-many" mapping. For this reason, instead of directly learning to predict continuous facial parameters, we frame the face generation under an NST framework by optimizing the input facial parameters to maximize the similarity between the created face and the real one.

3. Method

Our model consists of an Imitator $G(\boldsymbol{x})$ and a Feature Extractor $F(\boldsymbol{y})$, where the former aims to imitate the behavior of a game engine by taking in the user-customized facial parameters $\boldsymbol{x}$ and producing a "rendered" facial image $\boldsymbol{y}$, while the latter determines the feature space in which the facial similarity measurement can be performed to optimize the facial parameters. The processing pipeline of our method is shown in Fig. 2.

3.1. Imitator

We train a convolutional neural network as our imitator to fit the input-output relationship of a game engine so that to make the character customization system differentiable. We take the similar network configuration of DCGAN [33] in our imitator $G(\boldsymbol{x})$, which consists of eight transposed convolution layers. The architecture of our imitator is shown in Fig. 3. For simplicity, our imitator G only fits the front view of the facial model with the corresponding facial customization parameters.

We frame the learning and prediction of the imitator as a standard deep learning based regression problem, where we aim to minimize the difference between the in-game rendered image and the generated one in their raw pixel space. The loss function for training the imitator is designed as follows:

$$\begin{aligned}\mathcal{L}_G(\boldsymbol{x}) &= E_{\boldsymbol{x}\sim u(\boldsymbol{x})}\{\|\boldsymbol{y}-\hat{\boldsymbol{y}}\|_1\} \\ &= E_{\boldsymbol{x}\sim u(\boldsymbol{x})}\{\|G(\boldsymbol{x})-\text{Engine}(\boldsymbol{x})\|_1\},\end{aligned} \tag{2}$$

where $\boldsymbol{x}$ represents the input facial parameters, $G(\boldsymbol{x})$ represents the output of the imitator, $\hat{\boldsymbol{y}} = \text{Engine}(\boldsymbol{x})$ represents the rendering output of the game engine. We use l_1 loss function rather than l_2 as l_1 encourages less blurring. The input parameters $\boldsymbol{x}$ are sampled from a multidimensional uniform distribution $u(\boldsymbol{x})$. Finally, we aim to solve:

$$G^\star = \arg\min_G \mathcal{L}_G(\boldsymbol{x}). \tag{3}$$

In the training process, we randomly generate 20,000 individual faces with their corresponding facial customization

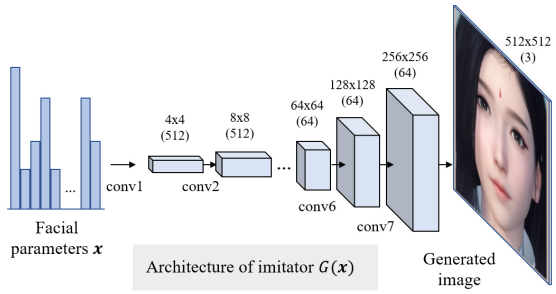

Figure 3. The architecture of our imitator $G(\boldsymbol{x})$. We train the imitator to learn a mapping from the input facial customization parameters $\boldsymbol{x}$ to the rendered facial image $\hat{\boldsymbol{y}}$ produced by the game engine.

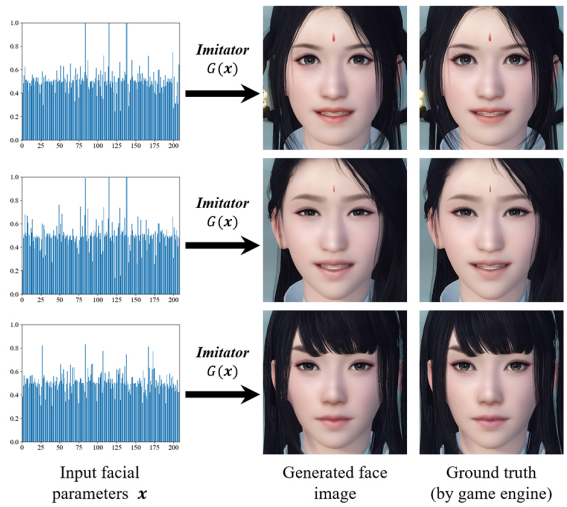

Figure 4. Some examples of the generated face images by our imitator $G(\boldsymbol{x})$ and the corresponding ground truth. The facial parameters of these images are created manually.

parameters by using the engine of the game "Justice". 80% face samples are used for training and the rests are used for validation. Fig. 4 shows three examples of the "rendering" results of our imitator. The facial parameters of these images are created manually. As the training samples are generated randomly from a unified distribution of facial parameters, it may look strange for most characters (please see our supplementary material). Nevertheless, as we can still see from Fig. 4 that, the generated face image and the rendered ground truth share a high degree of similarity, even in some regions with complex textures, e.g. the hair. This indicates that our imitator not only fits the training data in a low dimensional face manifold but also learns to decouple the correlations between different facial parameters.

3.2. Facial Similarity Measurement

Once we have obtained a well-trained imitator G, the generation of the facial parameters essentially becomes a face similarity measurement problem. As the input face photo and the rendered game character belong to different image domains, to effectively measure the facial similarity, we design two kinds of loss functions as measurements in terms of both global facial appearance and local details. Instead of directly computing their losses in raw pixel space, we take advantage of the neural style transfer frameworks and compute losses on the feature space that learned by deep neural networks. The parameter generation can be considered as a searching process on the manifold of the imitator, on which we aim to find an optimal point $\boldsymbol{y}^* = G(\boldsymbol{x}^*)$ that minimizes the distance between $\boldsymbol{y}$ and the reference face photo $\boldsymbol{y}_r$, as shown in Fig. 5.

3.2.1 Discriminative Loss

We introduce a face recognition model F_1 as a measurement of the global appearances of the two faces, e.g. the shape of the face and the overall impression. We follow the idea of the perceptual distance, which has been widely applied in a variety of tasks, e.g. image style transfer [12], super-resolution [21, 24], and feature visualization [45], and assume that for the different portraits of the same person, their features should have similar representations. To this end, we use a state of the art face recognition model "Light CNN-29 v2" [42] to extract the 256-d facial embeddings of the two facial images and then compute the cosine distance between them as their similarity. The loss function is referred as "discriminative loss" since it predicts whether the faces from a real photo and the imitator belong to the same person. The discriminative loss function of the above process is defined as follows:

$$\begin{aligned}\mathcal{L}_1(\boldsymbol{x}, \boldsymbol{y}_r) &= 1 - \cos(F_1(\boldsymbol{y}), F_1(\boldsymbol{y}_r)) \\ &= 1 - \cos(F_1(G(\boldsymbol{x})), F_1(\boldsymbol{y}_r)),\end{aligned} \tag{4}$$

where the cosine distance between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as:

$$\cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{<\boldsymbol{a}, \boldsymbol{b}>}{\sqrt{\|\boldsymbol{a}\|_2^2 \|\boldsymbol{b}\|_2^2}}. \tag{5}$$

3.2.2 Facial Content Loss

In additional to the discriminative loss, we further define a content loss by computing pixel-wise error based on the facial features extracted from a face semantic segmentation model. The facial content loss can be regarded as a constraint on the shape and displacement of different face components in two images, e.g. the eyes, mouth, and nose. As we care more about modeling facial contents rather than everyday images, the face semantic segmentation network is specifically trained to extract facial image features instead of using off-the-shelf models that are pre-trained on the ImageNet dataset [7]. We build our facial segmentation model

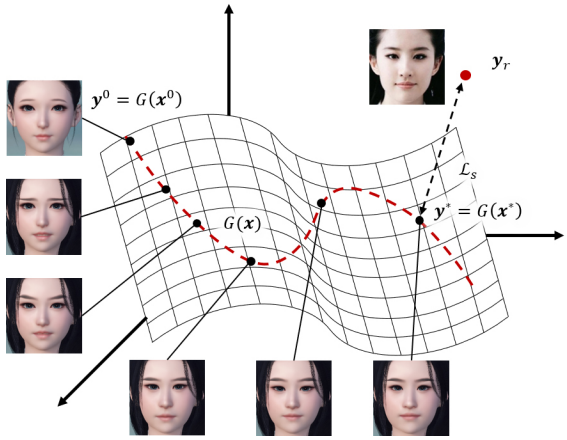

Figure 5. The game character auto-creation can be considered as a searching process on the manifold of the imitator. We aim to find an optimal point $\boldsymbol{y}^* = G(\boldsymbol{x}^*)$ that minimizes the distance between $\boldsymbol{y}$ and the reference face photo $\boldsymbol{y}_r$ in their feature space.

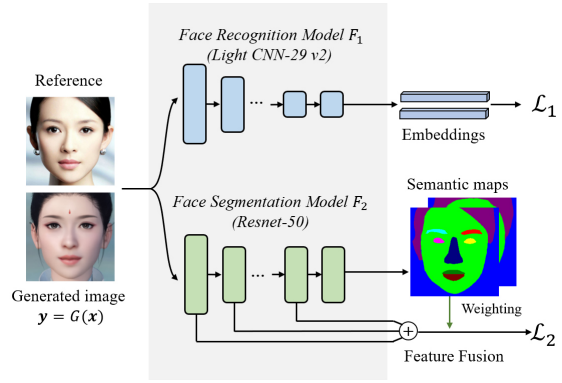

Figure 6. To effectively measure the similarity between two cross-domain face images, we design two kinds of loss functions, i.e. the discriminative loss $\mathcal{L}_1$ and the facial content loss $\mathcal{L}_2$, which are defined on the learned feature spaces.

based on the Resnet-50 [17] where we remove its fully connected layers and increase its output resolution from 1/32 to 1/8. We train this model on the well-known Helen face semantic segmentation dataset [23]. To improve the position sensitivity of the facial semantic feature, we further use the segmentation results (class-wise probability maps) as the pixel-wise weights of the feature maps to construct the position-sensitive content loss function. Our facial content loss is defined as follows:

$$\mathcal{L}_2(\boldsymbol{x}, \boldsymbol{y}_r) = \|\omega(G(\boldsymbol{x}))F_2(G(\boldsymbol{x})) - \omega(\boldsymbol{y}_r)F_2(\boldsymbol{y}_r)\|_1, \quad (6)$$

where F_2 represents the mapping from input image to the facial semantic features, ω represents the pixel-wise weights on the features, e.g., ω_1 is the eye-nose-mouth map.

The final loss function of our model can be written as a linear combination of the two objectives $\mathcal{L}_1$ and $\mathcal{L}_2$:

$$\begin{aligned}\mathcal{L}_S(\boldsymbol{x}, \boldsymbol{y}_r) &= \alpha\mathcal{L}_1 + \mathcal{L}_2 \\ &= \alpha(1 - \cos(F_1(G(\boldsymbol{x})), F_1(\boldsymbol{y}_r))) \\ &+ \|\omega(G(\boldsymbol{x}))F_2(G(\boldsymbol{x})) - \omega(\boldsymbol{y}_r)F_2(\boldsymbol{y}_r)\|_1,\end{aligned} \quad (7)$$

where the parameter α is used to balance the importance of two tasks. An illustration of our feature extractor is shown in Fig. 6. We use the gradient descent method to solve the following optimization problem:

$$\begin{aligned}&\min_{\boldsymbol{x}} \ \mathcal{L}_S(\boldsymbol{x}, \boldsymbol{y}_r) \\ &\text{s.t.} \ \ x_i \in [0, 1]\end{aligned} \quad (8)$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]$ represents the facial parameters to be optimized and $\boldsymbol{y}_r$ represents an input reference face photo. A complete optimization process of our method is summarized as follows:

- **Stage I.** Train the imitator G, the face recognition network F_1 and the face segmentation network F_2.
- **Stage II.** Fix G, F_1 and F_2, initialize and update facial parameters $\boldsymbol{x}$, until reach the max-number of iterations:
  $\boldsymbol{x} \leftarrow \boldsymbol{x} - \mu\frac{\partial\mathcal{L}_S}{\partial\boldsymbol{x}}$ (μ: learning rate).
  Project x_i to $[0, 1]$: $x_i \leftarrow \max(0, \min(x_i, 1))$.

3.3. Implementation Details

Imitator: In our imitator, the convolution kernel size is set to 4×4, the stride of each transposed convolution layer is set to 2 so that the size of the feature maps will be doubled after each convolution. The Batch-Normalization and ReLU activation are embedded in our imitator after every convolution layers, except for its output layer. Besides, we use the SGD optimizer for training with the batch_size $= 16$ and momentum $= 0.9$. The learning rate is set to 0.01, the learning rate decay is set to 10% per 50 epochs, and the training stops after 500 training epochs.

Facial segmentation network: We use Resnet-50 [17] as the backbone of our segmentation network by removing its fully connected layers and adding an additional 1×1 convolution layer at its top. Besides, to increase the output resolution, we change the stride from 2 to 1 at Conv_3 and Conv_4. Our model is pre-trained on the ImageNet [7], and then fine-tuned on the Helen face semantic segmentation dataset [23] with the pixel-wise cross entropy loss. We use the same training configurations as our imitator, except that the learning rate is set to 0.001.

Facial parameters: The dimension D of the facial parameters is set to 264 for "male" and 310 for "female". In these parameters, 208 of them are in continuous values (such as eyebrow length, width, and thickness) and the rest

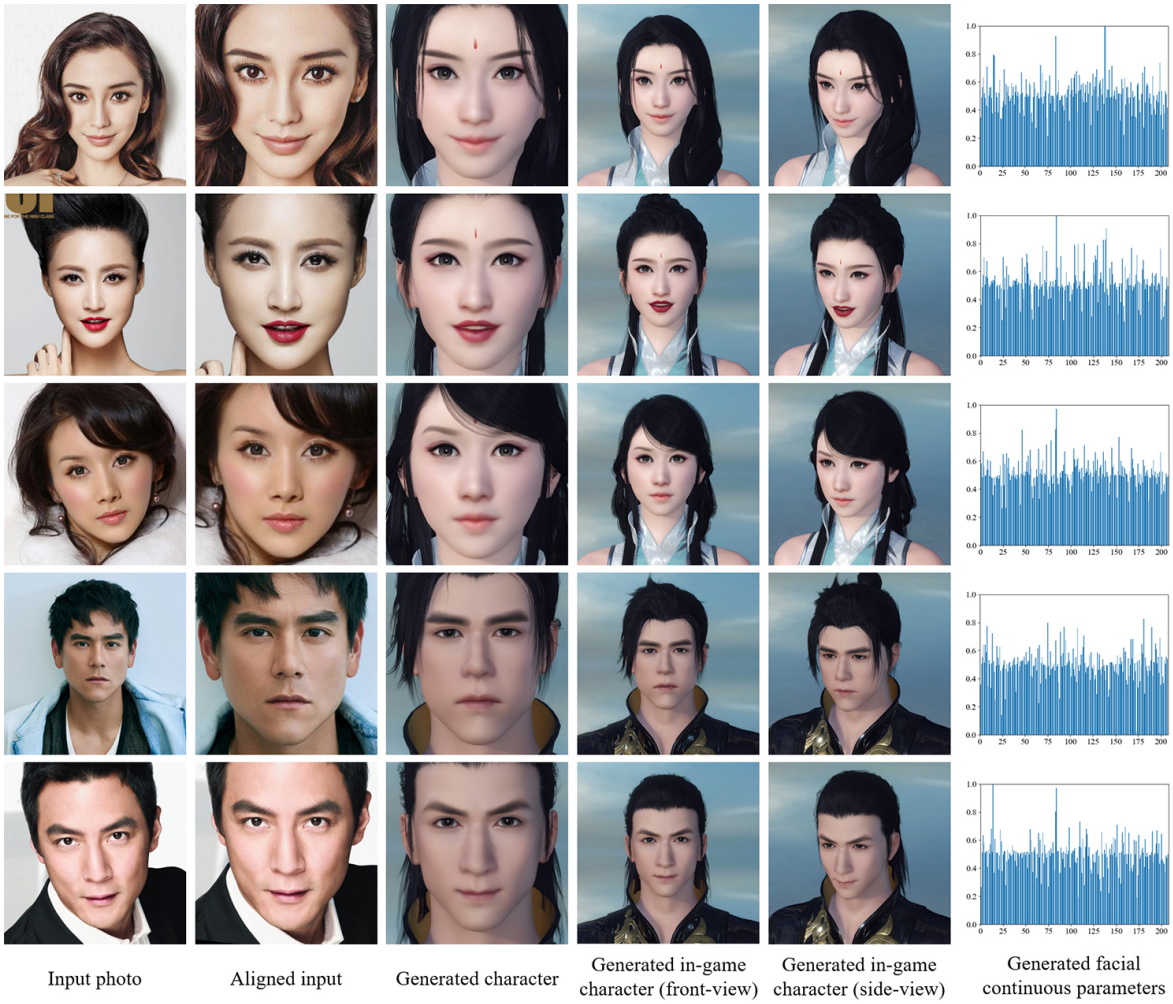

Figure 7. Some input photos and generated characters by using our method (both of the "identity" and "expressions" are modeled).

are discrete ones (such as hairstyle, eyebrow style, beard style, and lipstick style). These discrete parameters are encoded as one-hot vectors and are concatenated with the continuous ones. Since the one-hot encodings are difficult to optimize, we use the softmax function to smooth these discrete variables by the following transform:

$$h(\boldsymbol{x}, \beta) = \frac{e^{\beta x_k}}{\sum_{i=1}^{D'} e^{\beta x_i}} \quad k = 1, 2, \ldots, n, \tag{9}$$

where D' represents the dimension of discrete parameters' one-hot encoding. $\beta > 0$ controls the degree of smoothness. We set relatively large β, say, $\beta = 100$, to speed up optimization. We use "average face" to initialize the facial parameters $\boldsymbol{x}$, i.e. we set the all elements in continuous part as 0.5 and set those in discrete part as 0. For a detailed description of our facial parameters, please refer to our supplementary material.

Optimization: As for the optimization in Stage II, we set α as 0.01, the max-number of iterations as 50, the learning rate μ as 10 and its decay rate as 20% per 5 iterations.

Face alignment: The face alignment is performed (by using dlib library [22]) to align the input photo before it is fed into the feature extractor, and we use the rendered "average face" as its reference.

4. Experimental Results and Analysis

We construct a celebrity dataset with 50 facial close-up photos to conduct our experiments. Fig. 7 shows some input photos and generated facial parameters, from which an in-game character can be rendered by the game engine at multiple views and it shares a high degree of similarity to the input photo. For more generated examples, please refer to our supplementary material.

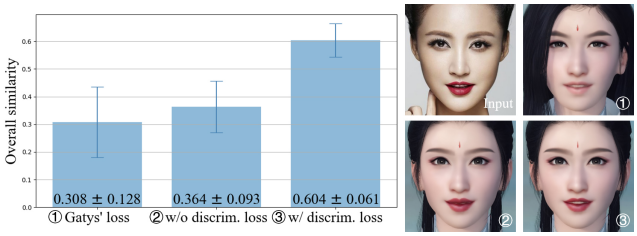

Figure 8. A performance comparison between different objective functions.

4.1. Ablation Studies

The ablation studies are conducted on our dataset to analyze the importance of each component of the proposed framework, including 1) discriminative loss and 2) facial content loss.

1) Discriminative loss. We run our method on our dataset w/ or w/o the help of discriminative loss and further adopt the Gatys' content loss [12] as the baseline. We compute the similarities between each photo and the corresponding generated results by using the cosine distance on the output of the face recognition model [42], as shown in Fig. 8. We can observe noticeable similarity improvement when we integrate the discriminative loss.

2) Facial content loss. Fig. 9 shows a comparison of the generated faces w/ or w/o the help the facial content loss. For a clearer view, the facial semantic maps and the edges of facial components are extracted. In Fig. 9, the yellow pixels of the edge maps correspond to the edge of the reference photo and the red pixels correspond to the generated faces. We can observe a better correspondence of the pixel location between the input photo and the generated face when we apply the facial content loss.

3) Subjective evaluation. To quantitatively analyze the importance of the two losses, we follow the subjective evaluation method used by Wolf *et al*. [41]. Specifically, we first generate 50 groups of character auto-creation results with different configurations of the similarity measurement loss functions ($\mathcal{L}_1$ only, $\mathcal{L}_2$ only and $\mathcal{L}_1+\mathcal{L}_2$) on our dataset. We then ask 15 non-professional volunteers to select the best work in each group, of which three characters are in random order. Finally, the selection ratio of an output character is defined as how many percentage it is selected by volunteers as the best one in its group, and the overall selection ratio is used to evaluate the quality of the results. The statistics are shown in Tab. 1, which indicates that both losses are beneficial for our method.

4.2. Comparison with other methods

We compare our method with some popular neural style transfer methods: global style method [12] and local style method [16]. Although these methods are not specifically designed for generating 3D characters, we still compare

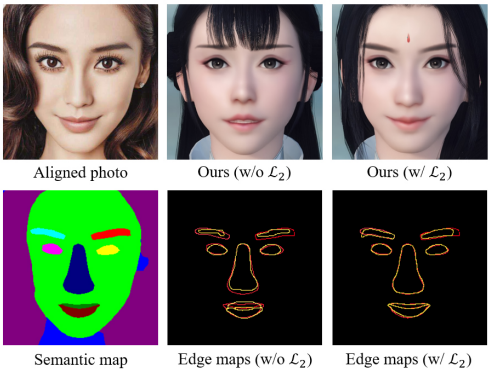

Figure 9. A comparison of the generated faces w/ or w/o the help of the facial content loss. The first column shows the aligned photo and its semantic map produced by our face segmentation model. The second and third columns show the generated faces w/ or w/o the facial content loss. Their edge maps are extracted for a better comparison, where the yellow pixels correspond to the edge of the reference photo and the red pixels correspond to the generated faces.

Ablations		
Discrim. $\mathcal{L}_1$	Facial-Sem. $\mathcal{L}_2$	Selection Ratio
✓	×	$13.47\% \pm 0.38\%$
×	✓	$36.27\% \pm 0.98\%$
✓	✓	$\mathbf{50.26\% \pm 0.40\%}$

Table 1. Subjective evaluation results of two technical components of our method 1) discriminative loss $\mathcal{L}_1$, 2) facial content loss $\mathcal{L}_2$ on our dataset. A higher selection ration indicates better.

with them since they are similar with our approach in multiple aspects. Firstly, these methods are all designed to measure the similarity of two images based on deep learning features. Secondly, the iterative optimization algorithms in these methods are all performed at the input of networks. As shown in Fig. 10, we can see that by separating the image styles from content and reorganizing them, it is difficult to generate vivid game characters. This is because the generated images are not exactly sampled from the game character manifold, thus it's hard to apply these methods in RPG environments. We also compare our method with a popular monocular 3D face reconstruction method: 3DMM-CNN [39], as shown in the right side of Fig. 10. Our auto-created game characters have a high similarity with the inputs while the 3DMM method can only generate masks with similar facial outlines.

To quantitatively evaluate the similarity between the generated face and the in-game style reference, we use the Mode Score (MS) [43] and the Fréchet Inception Distance (FID) [18] as our metrics. For each test image, we randomly select an image from the imitator training set as its reference and compute the average MS and FID over the

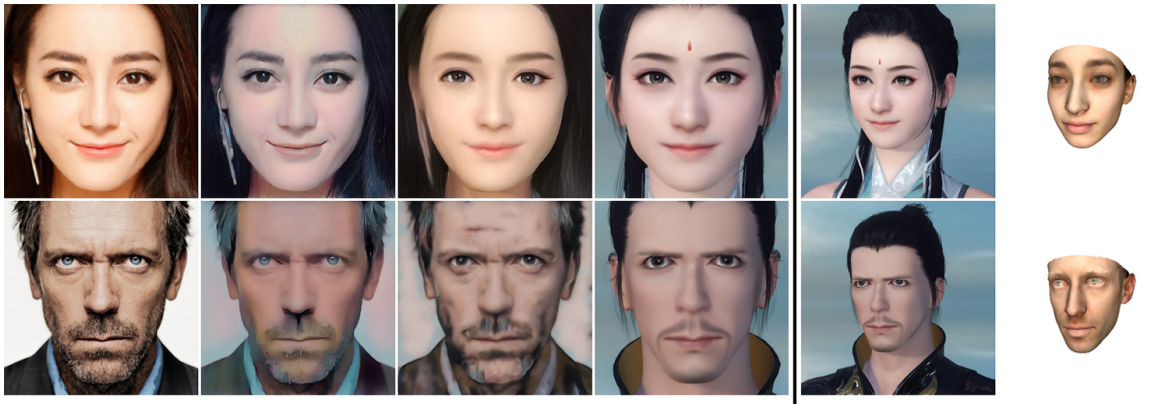

Figure 10. A comparison with other NST methods: Global style [12] and Local style [16], and we use the "average face" of each gender as the style reference of these NST methods. We also compare with a popular monocular 3D face reconstruction method: 3DMM-CNN [39].

Table 2. The style similarity and speed performance of different methods. (A higher Mode Score or a lower FID indicates better)

Method	Global style [12]	Local style [16]	3DMM-CNN [39]	Ours
Mode Score	1.0371 ± 0.0134	1.0316 ± 0.0128	–	$\mathbf{1.1418 \pm 0.0049}$
Fréchet Inception Distance	0.0677 ± 0.0018	0.0554 ± 0.0025	–	$\mathbf{0.0390 \pm 0.0018}$
Time (run on TITAN Xp)	22s	43s	15s	16s

entire test set. The above operations are repeated 5 times for computing the final mean value and the standard deviation as listed in Tab. 2. The runtime of each method is also recorded. Our method achieves higher style similarity and good speed performance compared with other methods.

4.3. Robustness and limitation.

We further evaluate our method on different blurring and illumination conditions, and our method proves to be robust to these changes, as shown in Fig. 11. The last group gives a failure case of our method. Since $\mathcal{L}_2$ is defined on local features, our method is sensitive to pose changes.

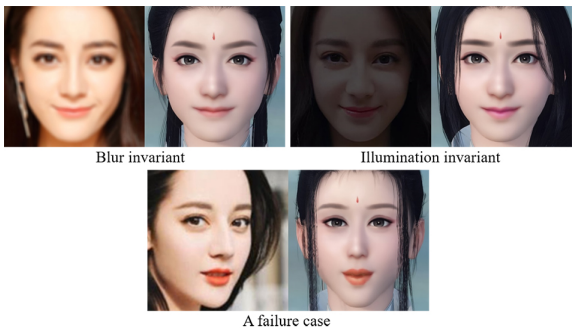

Figure 11. Experiments on robustness.

4.4. Generation with artistic portraits

Not limited to real photos, our method can also generate game characters for some artistic portraits, including the sketch image and caricature. Fig. 12 shows some examples of the generation results. Although the images are collected from a totally different distribution, we still obtain high-quality results since our method measures the similarity based on facial semantics rather than raw pixels.

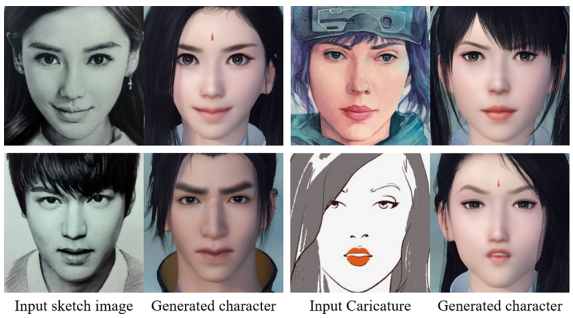

Figure 12. Game character auto-creation on artistic portraits.

5. Conclusion

In this paper, we propose a method for the automatic creation of an in-game character based on an input face photo. We formulate the creation under a facial similarity measurement and a parameter searching paradigm by solving an optimization problem over a large set of physically meaningful facial parameters. Experimental results demonstrate that our method achieves a high degree of generation similarity and robustness between the input face photo and the rendered in-game character in terms of both global appearance and local details.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Volker Blanz, Thomas Vetter, et al. A morphable model for the synthesis of 3d faces. In *Siggraph*, volume 99, pages 187–194, 1999.

[3] James Booth, Epameinondas Antonakos, Stylianos Ploumpis, George Trigeorgis, Yannis Panagakis, and Stefanos Zafeiriou. 3d face morphable models "in-the-wild". In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] Huiwen Chang, Jingwan Lu, Fisher Yu, and Adam Finkelstein. Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Pengfei Dou, Shishir K. Shah, and Ioannis A. Kakadiaris. End-to-end 3d face reconstruction with deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *Proc. of ICLR*, 2017.

[10] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM.

[11] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, 2017.

[12] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. Unsupervised training for 3d morphable model regression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Luthi, Sandro Schönborn, and Thomas Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 75–82. IEEE, 2018.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[16] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary style transfer with deep feature reshuffle. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc., 2017.

[19] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[22] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[23] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *European conference on computer vision*, pages 679–692. Springer, 2012.

[24] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[25] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems 30*, pages 386–396. 2017.

[27] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 36(4):120:1–120:15, jul 2017.

[28] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 469–477. Curran Associates, Inc., 2016.

[29] Ming Lu, Hao Zhao, Anbang Yao, Feng Xu, Yurong Chen,

and Li Zhang. Decoder network over lightweight reconstructed feature for fast semantic style transfer. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[30] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Attribute-guided face generation using conditional cyclegan. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[31] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[32] Weilong Peng, Zhiyong Feng, Chao Xu, and Yong Su. Parametric t-spline face morphable model for detailed fitting in shape subspace. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[33] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[34] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. Learning detailed face reconstruction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[35] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[36] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Trans. Graph.*, 36(4), July 2017.

[37] Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[38] Luan Tran and Xiaoming Liu. Nonlinear 3d face morphable model. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[39] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gerard Medioni. Regressing robust and discriminative 3d morphable models with a very deep neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[40] Anh Tuan Tran, Tal Hassner, Iacopo Masi, Eran Paz, Yuval Nirkin, and Gerard Medioni. Extreme 3d face reconstruction: Seeing through occlusions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[41] Lior Wolf, Yaniv Taigman, and Adam Polyak. Unsupervised creation of parameterized avatars. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[42] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light cnn for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896, 2018.

[43] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

[44] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[45] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

A. Appendix

A.1. Configurations of our networks

A detailed configuration of our Imitator G and Face Segmentation Network F_2 are listed in Table 3 and Table 4. As for the details of Face Recognition Network F_1, i.e. Light CNN-29 v2, please refer to Wu *et al*.'s paper [42].

Specifically, in a $c \times w \times w/s$ of Convolution / Deconvolution layer, c denotes the number of filters, $w \times w$ denotes the filter's size and s denotes the filter's stride. In a $w \times w/s$ of Maxpool layer, w denotes the pooling window size, and s denotes the pooling stride. In an n/s Bottleneck block [17], n denotes the number of planes, and s denotes the block's stride.

	Layer	Component	Configuration	Output Size
Imitator	Conv_1	Deconvolution + BN + ReLU	512x4x4 / 1	4x4
	Conv_2	Deconvolution + BN + ReLU	512x4x4 / 2	8x8
	Conv_3	Deconvolution + BN + ReLU	512x4x4 / 2	16x16
	Conv_4	Deconvolution + BN + ReLU	256x4x4 / 2	32x32
	Conv_5	Deconvolution + BN + ReLU	128x4x4 / 2	64x64
	Conv_6	Deconvolution + BN + ReLU	64x4x4 / 2	128x128
	Conv_7	Deconvolution + BN + ReLU	64x4x4 / 2	256x256
	Conv_8	Deconvolution	3x4x4 / 2	512x512

Table 3. A detailed configuration of our Imitator G.

	Layer	Component	Configuration	Output Resolution
Segmentation Model	Conv_1	Convolution + BN + ReLU	64x7x7 / 2	$\frac{1}{2} \times \frac{1}{2}$
	MaxPool	MaxPool	3x3 / 2	$\frac{1}{4} \times \frac{1}{4}$
	Conv_2	3 x Bottleneck	64 / 2	$\frac{1}{8} \times \frac{1}{8}$
	Conv_3	4 x Bottleneck	128 / 1	$\frac{1}{8} \times \frac{1}{8}$
	Conv_4	6 x Bottleneck	256 / 1	$\frac{1}{8} \times \frac{1}{8}$
	Conv_5	3 x Bottleneck	512 / 1	$\frac{1}{8} \times \frac{1}{8}$
	Conv_6	Convolution	11x1x1 / 1	$\frac{1}{8} \times \frac{1}{8}$

Table 4. A detailed configuration of our Face Segmentation Model F_2.

A.2. More examples of generated in-game characters

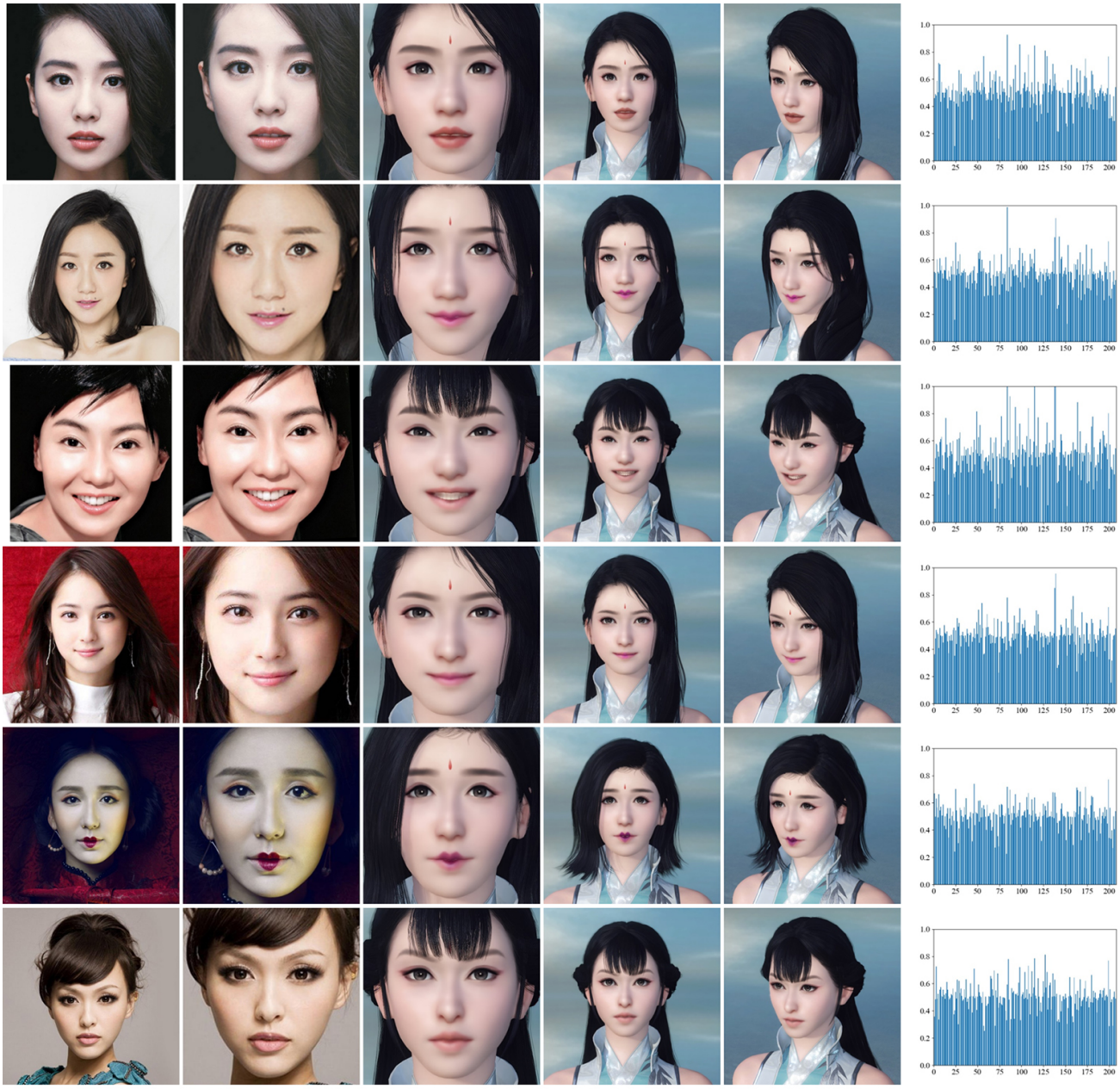

Figure 13. More generated in-game characters (female).

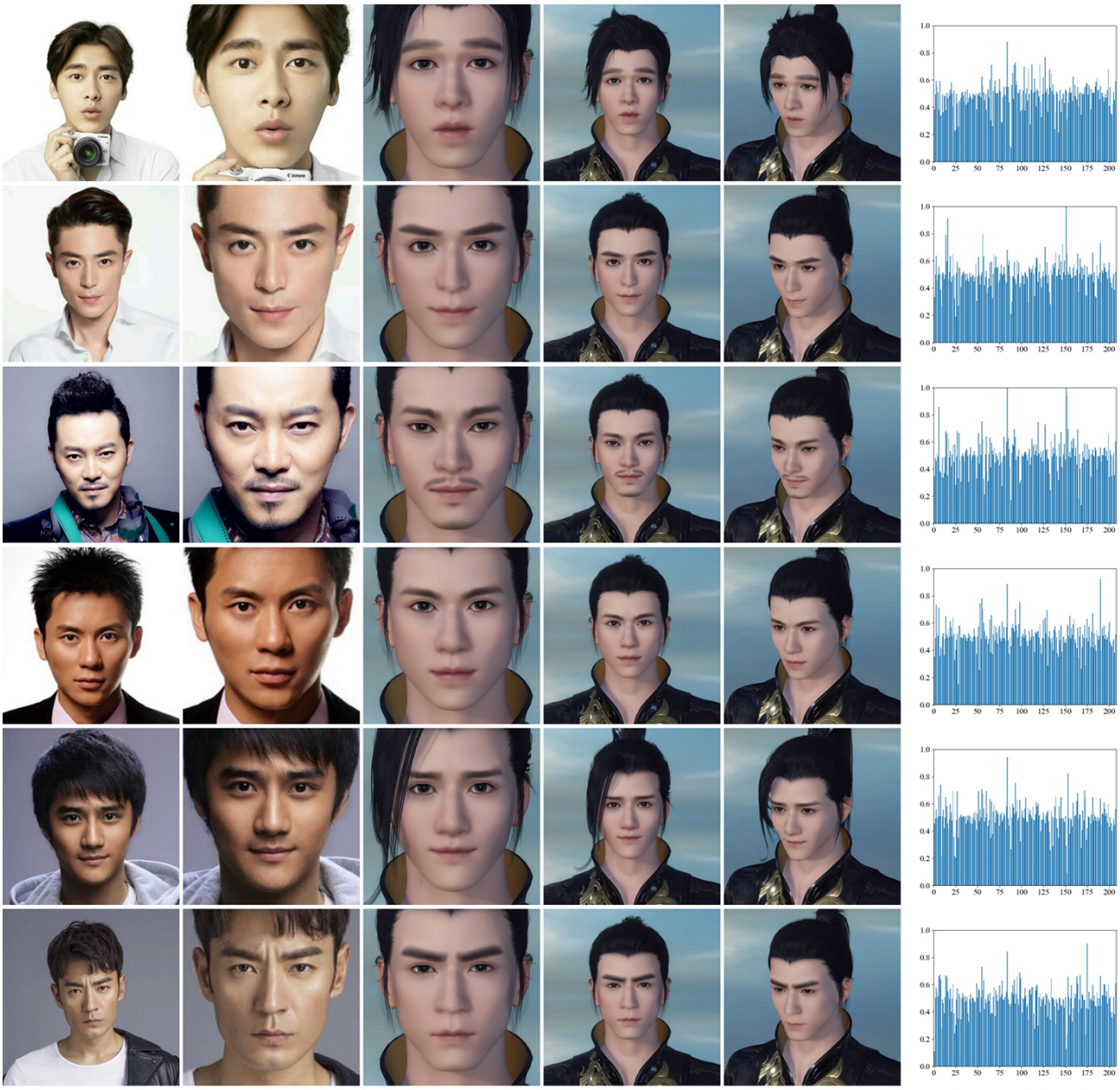

Figure 14. More generated in-game characters (male).

A.3. More comparison results

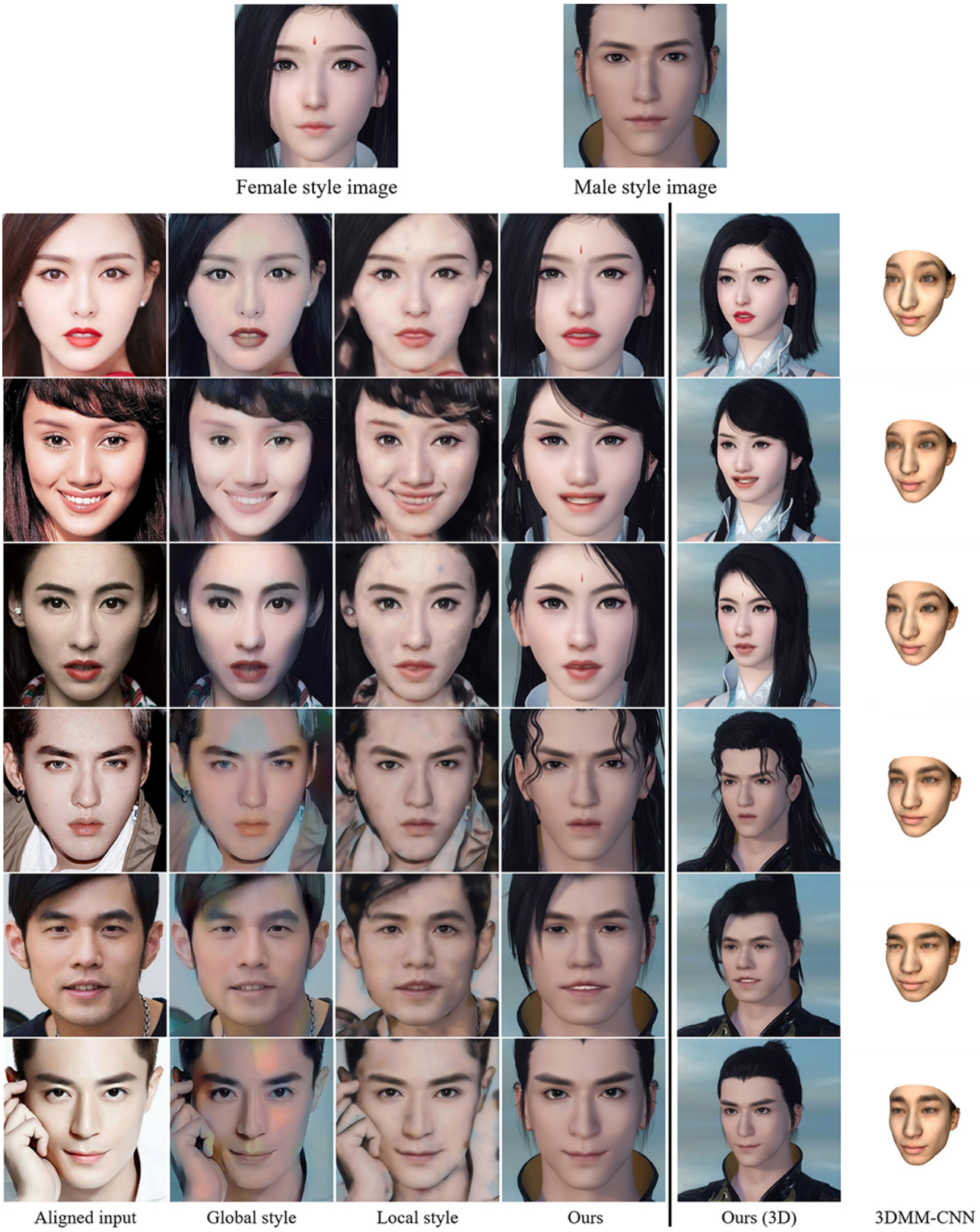

Figure 15. More comparison results with other NST methods: Global style [12] and Local style [16]. We use the "average face" of each gender as the style images. We also compare with a popular monocular 3D face reconstruction method: 3DMM-CNN [39].

A.4. Training samples of our Imitator

During the training process, we train our imitator with randomly generated game faces other than regular ones, as shown in Fig. 16. In our experiment, we adopt two imitators to fit female and male 3D models respectively, in order to auto-create characters for different genders.

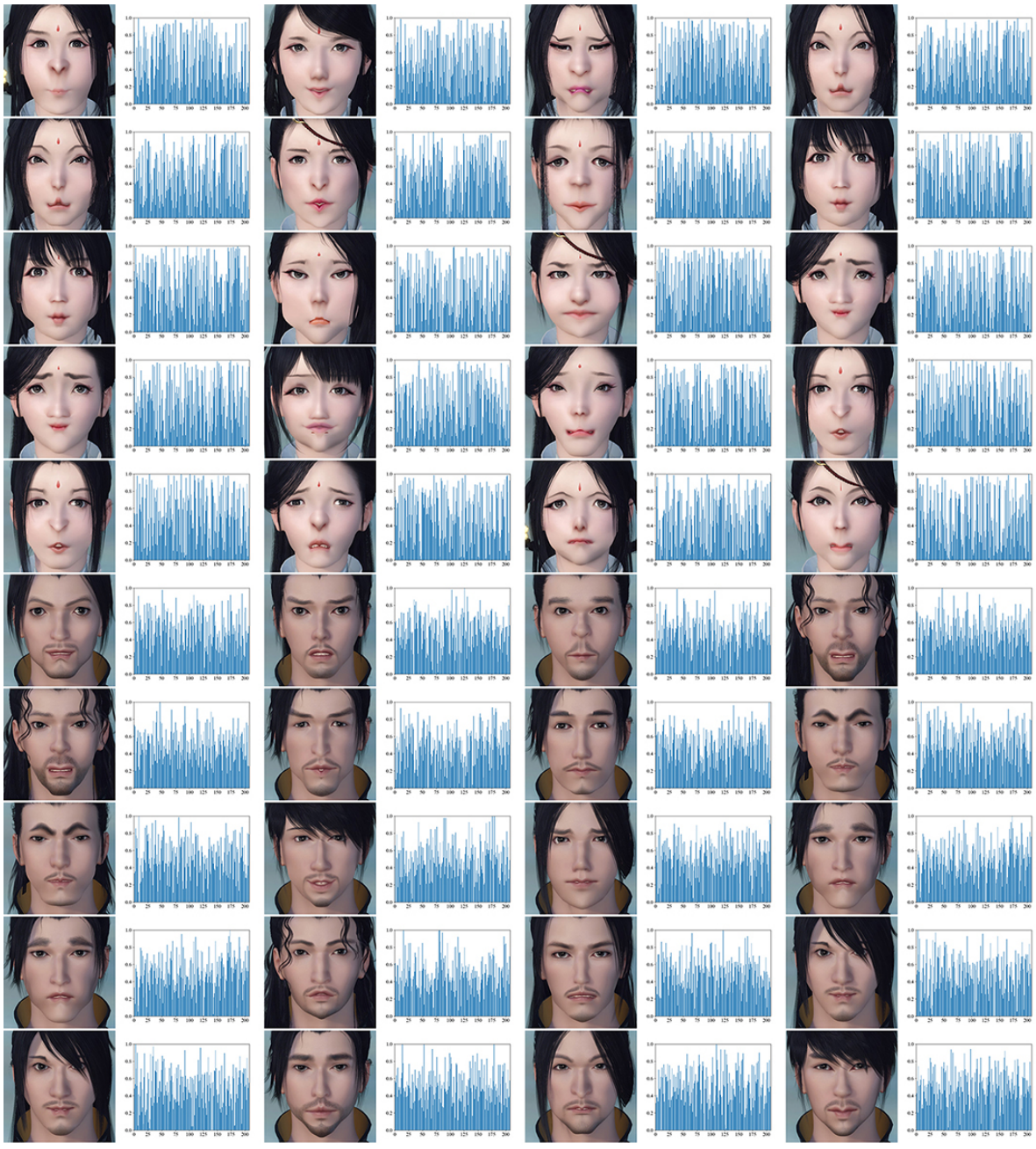

Figure 16. Training samples of imitator and the corresponding facial parameters.

A.5. A description of facial parameters

Table 5 lists a detailed description of each facial parameter, where the "Component" represents the facial parts which parameters belong to, the "Controllers" represents user-adjustable parameters of each facial part (one controller correspond to one continuous parameter), and the "# Controllers" represents the total number of controllers, i.e. 208. Besides, there are additional 102 discrete parameters for female (22 hair styles, 36 eyebrow styles, 19 lipstick styles, and 25 lipstick colors) and 56 discrete parameters for male (23 hair styles, 26 eyebrow styles, and 7 beard styles).

Component		Controllers	# controllers	Sum
Eyebrow	eyebrow-head	horizontal-offset, vertical-offset, slope, ...	8	208
	eyebrow-body	horizontal-offset, vertical-offset, slope, ...	8	
	eyebrow-tail	horizontal-offset, vertical-offset, slope, ...	8	
Eye	whole	horizontal-offset, vertical-offset, slope, ...	6	
	outside upper eyelid	horizontal-offset, vertical-offset, slope, ...	9	
	inside upper eyelid	horizontal-offset, vertical-offset, slope, ...	9	
	lower eyelid	horizontal-offset, vertical-offset, slope, ...	9	
	inner corner	horizontal-offset, vertical-offset, slope, ...	9	
	outer corner	horizontal-offset, vertical-offset, slope, ...	9	
Nose	whole	vertical-offset, front-back, slope	3	
	bridge	vertical-offset, front-back, slope, ...	6	
	wing	horizontal-offset, vertical-offset, slope, ...	9	
	tip	vertical-offset, front-back, slope, ...	6	
	bottom	vertical-offset, front-back, slope, ...	6	
Mouth	whole	vertical-offset, front-back, slope	3	
	middle upper lip	vertical-offset, front-back, slope, ...	6	
	outer upper lip	horizontal-offset, vertical-offset, slope, ...	9	
	middle lower lip	vertical-offset, front-back, slope, ...	6	
	outer lower lip	horizontal-offset, vertical-offset, slope, ...	9	
	corner	horizontal-offset, vertical-offset, slope, ...	9	
Face	forehead	vertical-offset, front-back, slope, ...	6	
	glabellum	vertical-offset, front-back, slope, ...	6	
	cheekbone	horizontal-offset, vertical-offset, slope, ...	5	
	risorius	horizontal-offset, vertical-offset, slope, ...	5	
	cheek	horizontal-offset, vertical-offset, width, ...	6	
	jaw	vertical-offset, front-back, slope, ...	6	
	lower jaw	horizontal-offset, vertical-offset, slope, ...	9	
	mandibular corner	horizontal-offset, vertical-offset, slope, ...	9	
	outer jaw	horizontal-offset, vertical-offset, slope, ...	9	

Table 5. A detailed interpretation of each facial parameter (continuous part).