Neural Rendering for Game Character Auto-creation

Tianyang Shi†, Zhengxia Zou†, Zhenwei Shi and Yi Yuan*

Abstract—Many role-playing games feature character creation systems where players are allowed to edit the facial appearance of their in-game characters. This paper proposes a novel method to automatically create game characters based on a single face photo. We frame this "artistic creation" process under a self-supervised learning paradigm by leveraging the differentiable neural rendering. Considering the rendering process of a typical game engine is not differentiable, an "imitator" network is introduced to imitate the behavior of the engine so that the in-game characters can be smoothly optimized by gradient descent in an end-to-end fashion. Different from previous monocular 3D face reconstruction which focuses on generating 3D mesh-grid and ignores user interaction, our method produces fine-grained facial parameters with a clear physical significance where users can optionally fine-tune their auto-created characters by manually adjusting those parameters. Experiments on multiple large-scale face datasets show that our method can generate highly robust and vivid game characters. Our method has been applied to two games and has now provided over 10 million times of online services.

Index Terms—Game character customization, role-playing games, neural rendering, deep learning.

1 Introduction

The character customization system in many Role-Playing Games (RPGs) provides an interactive interface for players to edit the facial appearance of the in-game characters with their preferences instead of using default templates. To improve the player's immersion and interactivity, character customization systems are becoming sophisticated - in many modern RPGs such as "Grand Theft Auto Online" (https://www.rockstargames.com/GTAOnline) and "Dark Souls III" (https://www.darksouls.jp), players are now allowed to precisely manipulate their characters on detailed parts, e.g., corner of the eyes, hairstyles, and even makeups. As a result, the character customization process turns out to be laborious and time-consuming for most players. To create an in-game character with a desired facial appearance (e.g. a pop star or the players themselves), most players need to spend hours manually adjusting hundreds of parameters, even after considerable practice.

In computer vision, efforts have been made in generating 3D faces based on a single input face photo, in which a representative group of methods are the 3D Morphable Model (3DMM) [1] and its variants [2–4]. However, these methods are difficult to be applied to in-game environments due to the style gap and the different infrastructure between the two environments. In this paper, we propose a novel method to automatically create in-game characters for players according to a single input face photo[1], as shown in Fig. 1. We frame this "artistic creation" process under a self-supervised learning paradigm by leveraging the differentiable neural rendering. Different from the 3DMM based methods which focus on generating 3D mesh-grid and ignore user interaction, our method produces fine-grained facial parameters with a clear physical significance where users can optionally fine-tune their auto-created characters by manually adjusting those parameters according to their needs. We refer to our methods as a "Face-to-Parameter" translation method. In our method, each facial parameter controls the attribute (e.g., the position, orientation, and scale) of an individual facial component. As the rendering process of a typical game engine is not differentiable, a generative network G is designed as an "imitator" to im-

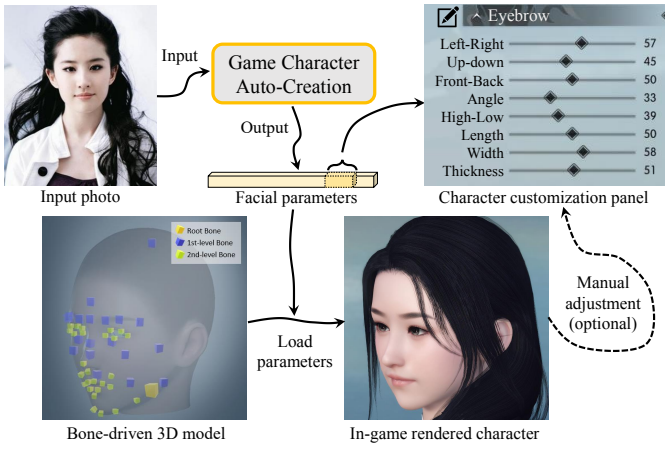


Fig. 1. We propose a novel method for game character auto-creation under a self-supervised learning framework by leveraging differential neural rendering. The proposed method converts a single input face photo to a large set of physically meaningful facial parameters. Users can further fine-tune the parameters optionally according to their needs.

† *these authors contributed equally to this work*
* *corresponding author: yuanyi@corp.netease.com*

- *Tianyang Shi and Yi Yuan are with Fuxi AI Lab, Netease, Hangzhou, Zhejiang 310052, China.*
- *Zhengxia Zou is with Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA.*
- *Zhenwei Shi is with Image Processing Center, School of Astronautics, Beihang University, Beijing 100191, China.*

1. Preliminary versions of this work were published in ICCV 2019 [5] and AAAI 2020 [6].

itate the physical behavior of the game engine so that our model can be learned by gradient descent in an end-to-end fashion. By taking advantage of the differentiable rendering in our method, the character auto-creation can be naturally formulated as a cross-domain facial similarity measurement problem between the face of a generated character and a real one. The training of our framework neither requires any ground truth references nor any user interactions.

Fig. 2 (a) shows an overview of the proposed method. Our method consists of multiple components:

- **An imitator** G. We design an imitator G to imitate the behavior of a game engine and make the rendering process differentiable, as shown in Fig. 2 (b). The G takes in a group of facial customization parameters and is trained to produce the face images of the corresponding in-game character. Fig. 2 (c) shows two examples of the "rendering" output of our imitator and their in-game "ground truth".

- **A translator** T. We introduce a facial parameter translator T, which aims to transform an input facial image to a set of in-game facial parameters. The generated parameters can be either used for rendering 3D characters in the game environments or further manually fine-tuned by players.

- **A facial descriptor** F. We take advantage of the deep Convolutional Neural Networks (CNNs) and introduce a facial descriptor F which learns high-level facial representations in terms of both global facial identity and local details. The descriptor consists of two networks, a face recognition network F_{recg}, and a face segmentation network F_{seg}, where the former encodes an input face image to a set of pose-irrelevant face embeddings and the latter learns position-sensitive face representations.

We formulate the training of our method as a multi-task regression process with multiple self-supervised loss functions. To measure the similarity between the input face and the generated one, we define two loss functions, an "identity loss" $\mathcal{L}_{idt}$ and a "facial content loss" $\mathcal{L}_{ctt}$, where the former one focuses on facial identity (pose-irrelevant) and the later one compute facial similarity base on pixel-wise representations. To improve the robustness and stability of the generation, we further introduce a "loopback loss" $\mathcal{L}_{loop}$, which ensures the translator correctly interprets its own output [3]. By minimizing the distance between the created face and the real one, the input face photos can be effectively converted to vivid in-game characters. On basis of the above framework, we propose two generation modes for facial parameters, a "one-shot" generation mode and an "iterative" generation mode, where the former one generates the facial parameters directly from the input image through the translator T in a single forward propagation while in the latter one we discard the translator and frame the generation as a parameter searching process at the input end of the renderer. We show in different aspects of their advantages in our experiment, such as high-quality facial generation, robustness, and speed. Our method has been applied to two role-playing games, a PC game "Justice" (in October 2018, https://n.163.com) and a mobile game "Heaven" (coming soon, https://tym.163.com/) and now has provided over 10 million times of online services.

Our contributions are summarized as follows:

- We propose a novel method for game character auto-creation. To our best knowledge, we are the first to launch this feature in the gaming industry.
- Since the rendering process of mainstream game engines is not differentiable, we introduce an imitator by building a deep generative network to imitate the behavior of the engine. In this way, the gradient of the facial similarity can be smoothly back-propagated all-though the generating pipeline and the model can be optimized in an end-to-end fashion.
- We introduce multiple loss functions under a self-supervised learning paradigm which proves to be effective for cross-domain facial similarity measurement. The loss functions can be jointly optimized by multi-task learning.

2 Related Work

2.1 Monocular 3D face reconstruction

Recovering 3D information from a single 2D face image has long been a challenging but important task in computer vision. On one hand, it forms the foundation of a large group of real-world applications, such as game production, medical plastic surgery, facial augmented reality, and virtual reality, etc. On the other hand, it is a typical ill-posed problem where the difficulty lies not only in the missing of the stereoscopic information but also in a highly variable imaging environment such as illumination changes, occlusion, blurring, etc.

A representative of early monocular 3D face reconstruction is the 3D Morphable Model (3DMM), which was originally proposed by Blanz *et al*. in 1999 [1]. In 3DMM and its recent variants [1, 7–13], a 3D mesh of a morphable face model is first parameterized and then optimized to fit the projection of the model to the 2D input face. In recent years, deep CNNs [14, 15] were introduced to the monocular 3D face reconstruction by taking advantage of the high-level image representations [2, 16–21]. These methods typically formulate the reconstruction as a standard regression problem between the 2D input and the morphable model. The regression based paradigm allows the integration of auxiliary constraints on their objectives such as the adversarial loss [22] and the loopback loss to achieve high-fidelity reconstruction results [3, 4].

2.2 Differentiable rendering

Graphic rendering is a fundamental problem in computer graphics that converts 3D models into 2D images. Traditional rendering pipelines used in 3D graphics consider the forward process only. These methods typically involve a discrete operation called rasterization, which prevents gradient back-propagation and thus makes the renderer non-differentiable. Differentiable Rendering (DR), which allows calculation of the derivative from the rendering output to the input 3D model, camera parameters, and even environment variables (e.g., light conditions), has drawn increasing attention in recent years. The key to the DR is to approximate the gradient of the rendering process by using a set of differentiable operators or structural units. The first differentiable renderer, OpenDR [23], was proposed by M. Loper *et al*. in 2014. In their method, they use the first-order Taylor expansion, i.e., a series of predefined spatial filters to

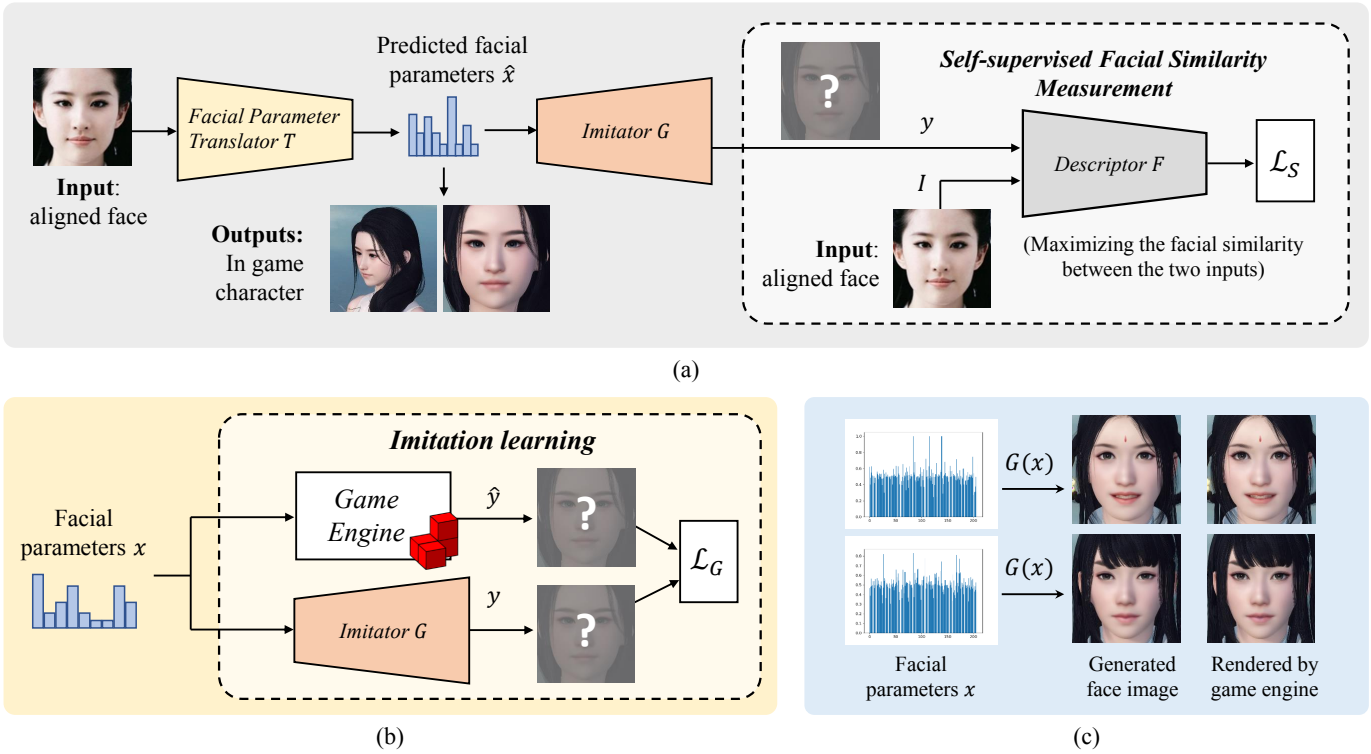


Fig. 2. An overview of our method. (a) Our model consists of multiple components: an imitator G, a facial parameter translator T, and a facial descriptor F. The G, as shown in (b), aims to imitate the behavior of a game engine by taking in a set of user-customized facial parameters x and producing a "rendered" facial image y. The T is trained to convert an input face photo to facial parameters which maximize the facial similarity between the input and the "rendering" result in the feature space produced by the F. In (c), we show two groups of faces generated by our imitator and their in-game ground truth.

approximate the gradient of the rasterizer. Later, some other approaches based on interpolation approximation [24, 25], triangle barycentric interpolation [3], and Monte Carlo [26] were proposed to improve the fidelity rendering result as well as the accuracy of the gradients. Since deep neural networks provide naturally differentiable topology, a new research topic called "neural rendering" quickly emerged and the neural networks were introduced to the rendering tasks [27–29].

As the 3D face reconstruction can be essentially considered as a parameter fitting problem between pre-scanned 3D faces and input facial image, making renderer differentiable recently became the key to solve this problem. Thanks to the development of the differentiable rendering and self-supervised learning, neural networks are now able to achieve high-fidelity rendering results even without using pre-scanned 3D faces [3, 19]. Despite the recent advances in this field, most of the 3D face reconstruction methods are not naturally applicable to RPGs. The main reason behind this is that the face models in these methods are typically designed based on statistical (PCA) shape basis but the statistical shape basis is not friendly for user interactions. As a comparison, the 3D faces in most RPGs are usually represented by using manually defined shape basis (e.g., bone-driven face models) in which the parameters have clear semantics. Table 1 shows a comparison between the two groups of representation methods.

Beyond the above morphable face model and bone-driven face model, Gruber *et al*. recently proposed a new anatomical local face model for interactive sculpting [30]. In their method, the authors integrate anatomical knowledge into the 3D scanned face models to obtain a high degree of freedom for face editing. Note that since our proposed neural renderer – the imitator, is not limited to the choice of the face model, our method can also be well applied on either of the morphable face model or the anatomical local face model. However, considering that the bone-driven face model is more frequently used in Massively Multi Player On-line Role-Playing Games (MMO-RPGs) than other types of face models, we only focus on the bone-driven face models in this paper.

TABLE 1
A comparison between "Statistical Shape Basis (SSB)" and "Manually Defined Shape Basis (MDSB)" on 3D face representation.

	SSB	MDSB
Semantics	ambiguous	explicit
Flexibility	normal	high
Texture style	real	real or game-style
Ground truth	3D scans	none
Makeup	a few	many (in-game)
Face model	morphable face model	bone-driven face model

3 Methodology

Here we introduce each part of our method in details, including our imitator G, facial parameter translator T, our

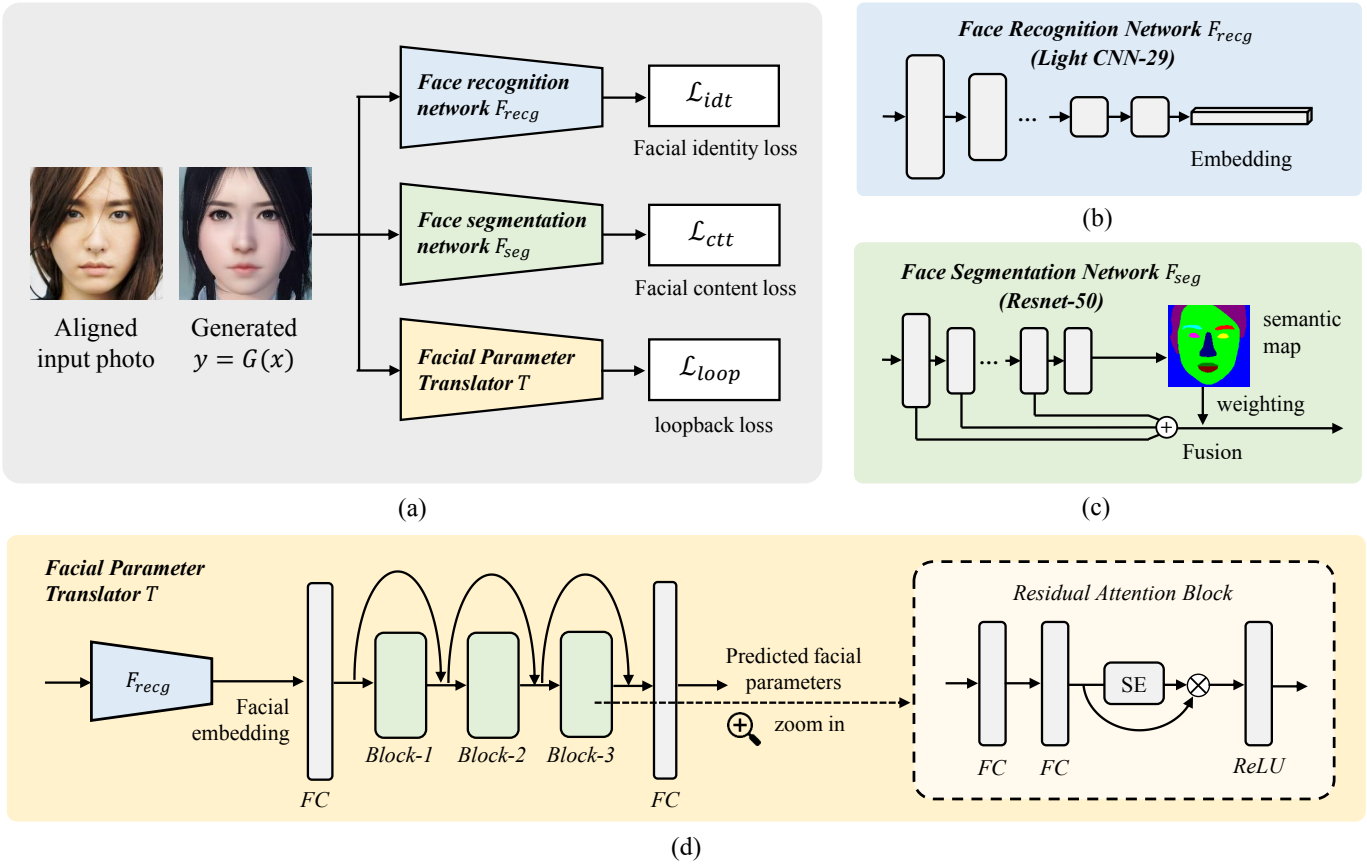


Fig. 3. (a) To effectively measure the cross-domain similarity between a generated face and a real one, we design three loss functions: 1) a facial identity loss $\mathcal{L}_{idt}$, which computes the distance between two images on top of the facial embeddings produced by a pre-trained face recognition network F_{recg}, as shown in (b); 2) a facial content loss $\mathcal{L}_{ctt}$, which computes the pixel-wise similarity based on the facial semantic maps produced by a face segmentation network F_{seg}, as shown in (c); 3) a loopback loss $\mathcal{L}_{loop}$, which ensures the facial parameter translator T correctly interprets its own output. (d) shows the architecture of our translator T. The key to our method is a self-supervised learning framework where a "recursive consistency" is introduced to *enforce* the facial representation of the rendered image I' to be similar with the input I: $I' = G(T(I)) \approx I$.

facial similarity measurement, and other implementation details.

3.1 Imitator

We train a convolutional neural network with eight transposed convolution layers as our imitator G to learn the behavior of a game engine so as to make the character customization system differentiable. We take the similar network configuration of the DCGAN [31] for our imitator. Fig. 4 shows an illustration of our imitator.

We frame the "rendering" as a standard pixel-wise regression problem, where we aim to minimize the difference between the in-game rendered image and the generated one in their raw pixel space. We train our imitator to minimize the following objective function:

$$\mathcal{L}_G(x) = E_{x\sim u(x)}\{\|G(x) - \text{Engine}(x)\|_1\}, \quad (1)$$

where x is the facial parameters sampled from a uniform distribution. Given a group of input parameters x, $G(x)$ and Engine(x) represent the outputs of our imitator and the game engine (ground truth), respectively. We use the pixel-wise l_1 loss rather than l_2 since the l_1 encourages less blurring effect. In the training process, we randomly generate 20,000 faces as well as their corresponding facial

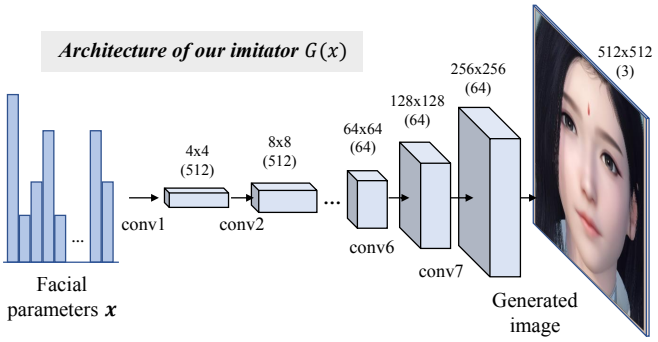


Fig. 4. The architecture of our imitator $G(x)$. We train the imitator to learn a mapping from a group of facial customization parameters x to a rendered facial image $\hat{y}$ produced by the game engine.

customization parameters for each game by using the game engine. For simplicity, our imitator G only fits the front view images of the facial model. An advantage of using neural networks for rendering is that it can handle complex lighting/shading models under a unified framework. This is a reason why we train a CNN as our differentiable renderer in our method.

3.2 Translator

In our facial parameter translator T, we train a neural network T' to map the facial embeddings to our in-game facial parameters x:

$$x = T(I) = T'(F_{recg}(I)), \tag{2}$$

where I is an input face photo and $f_{recg} = F_{recg}(I)$ is the facial embeddings produced by the face recognition network F_{recg}. Since the embeddings correspond to a global description of the facial identity while the facial parameters depict local details, to learn better correspondence of the two fields, we introduce an attention-based block as the core building block of our translator. We use three residual attention blocks and two Fully Connected (FC) layers on top of the facial embeddings to translate the embeddings to facial parameters. Fig. 3 (d) shows the details of our translator.

In each of our attention-based block, we compute the element-wise importance of the neurons and then performs feature re-calibration by multiplying the representations with them. We make a simple modification of the squeeze and excitation block in SENet [32] to apply it to an FC layer (the global pooling layer thus is removed). Suppose $v \in \mathbb{R}^{c\times 1}$ represents a c dimensional internal representation of our translator. We use a gating function to learn a group of attention weights $\alpha = \sigma(W_2\delta(W_1 v))$, where $\sigma(\cdot)$ and $\delta(\cdot)$ denote the sigmoid and ReLU activation function, respectively. W_1 and W_2 are the weights of two FC layers. The internal representation v is element-wisely re-scaled by the attention weights $\widetilde{v}_k = \alpha_k v_k$, $k = 1, \dots c$.

3.3 Facial Similarity Measurement

Once we have a well-trained imitator G, the generation of the facial parameters finally becomes a face similarity measurement problem. The measurement is conducted under a self-supervised learning framework, i.e., to *enforce* the facial representation of the rendered image I' to be similar to that of its input face photo I:

$$I' = G(T(I)) \approx I. \tag{3}$$

As the input face photo and the rendered game character belong to different image domains, to effectively measure the facial similarity, we design three types of loss functions as measurements - a facial identity loss $\mathcal{L}_{idt}$, a facial content loss $\mathcal{L}_{ctt}$, and a loopback loss $\mathcal{L}_{loop}$, as shown in Fig. 3 (a). The final loss function in our model can be written as the summary of the above three losses:

$$\mathcal{L}(G, T, F_{recg}, F_{seg}) = \lambda_1\mathcal{L}_{idt} + \lambda_2\mathcal{L}_{ctt} + \lambda_3\mathcal{L}_{loop}, \tag{4}$$

where $\lambda_i > 0$, $i = 1, 2, 3$ control the balance between different loss terms.

Facial identity loss: We use a popular face recognition network named LightCNN-29v2 [33] to conduct measurement of the global appearances of the two faces, as shown in Fig 3 (b). We follow the idea of perceptual distance, which has been widely applied in a variety of tasks, e.g. image style transfer [34], super-resolution [35, 36], and feature visualization [37], and assume that for the different portraits of the same person, their features should have similar representations. We use the backbone of the the LightCNN-29v2 to compute a 256-d face embedding and define the facial identity loss between two faces as the cosine distance on their embeddings:

$$\mathcal{L}_{idt} = 1 - e_1^T e_2 / \sqrt{\|e_1\|_2^2 \|e_2\|_2^2}, \tag{5}$$

where $e_1 = F_{recg}(I)$, $e_2 = F_{recg}(I')$ are the face embeddings of an input face photo I and a rendered face image I'.

Facial content loss: In addition to the facial identity loss, we also define a content loss by computing pixel-wise image distance based on the facial representations extracted from a pre-trained face semantic segmentation model F_{seg}, as shown in Fig 3 (c). The facial content loss provides constraints on the contour and displacement of different face components in two images regardless of different image domains. We build our face segmentation model based on Resnet-50 [15]. 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. We define the facial content loss as follows:

$$\mathcal{L}_{ctt} = \|\omega_1 f_1 - \omega_2 f_2\|_1, \tag{6}$$

where k is the pixel location of the feature map. $f_1 = F_{seg}(I)$, $f_2 = F_{seg}(I')$ are the facial semantic features of the image I and I'. ω_1 and ω_2 are the class-wise probability maps of facial components.

Loopback loss: Inspired by the unsupervised 3D face reconstruction method proposed by Genova *et al*. [3], we also introduce a "loopback loss" to further improve the robustness of our prediction. After we obtain the rendered face image I', we further feed it into our translator T to produce a set of new parameters $x' = T(I')$ and force the generated facial parameters before and after the loop unchanged, as shown in Fig 3 (d). The loopback loss can be defined as follows:

$$\mathcal{L}_{loop} = \|x - T(I')\|_1. \tag{7}$$

3.4 One-shot generation and iterative generation

We propose two types of methods for facial parameter generation under our facial similarity measurement framework - 1) one-shot generation (our default method) and 2) iterative generation.

3.4.1 One-shot generation

In the one-shot generation mode, the facial parameters are directly generated from the input photo I by passing through a fully trained translator $T^\star$: $x^\star = T^\star(I)$. In the training phase, the G, F_{recg} and F_{seg} are first trained separately. Then we fixed their weights and train our translator T by minimizing the facial similarity loss function (4)

$$\begin{aligned} T^\star &= \arg\min_T \ \mathcal{L}(G, T, F_{recg}, F_{seg}) \\ &= \arg\min_T \ (\lambda_1\mathcal{L}_{idt} + \lambda_2\mathcal{L}_{ctt} + \lambda_3\mathcal{L}_{loop}). \end{aligned} \tag{8}$$

A detailed optimization pipeline of the one-shot generation is summarized as follows:

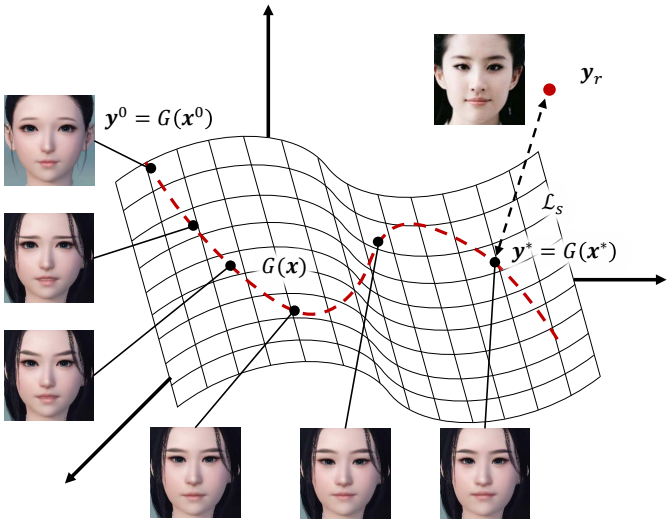


Fig. 5. In our iterative generation mode, the game character auto-creation can be considered as a searching process on the manifold of the imitator. We aim to find an optimal point $y^* = G(x^*)$ that minimizes the distance between y and the reference face photo y_r in their feature space.

- **Stage I** (training). Train the imitator G, face recognition network F_{recg} and the face segmentation network F_{seg} separately.
- **Stage II** (training). Fix G, F_{recg}, F_{seg}, and train the translator T.
- **Stage III** (inference). Given an input photo I, predict the facial parameters by using the well-trained translator $T^\star$: $x = T^\star(I)$.

We use the CelebA dataset [38] to train our translator T. We freeze all other networks (F_{recg}, F_{seg}, and T) when training our translator, and set $\lambda_1 = 0.01$, $\lambda_2 = 1$, and $\lambda_3 = 1$. We use the Adam optimizer [39] with the learning_rate = 10^{-4} and max-iteration = 20 epochs. To improve the prediction on side-view faces, we set $\lambda_2 = 0$ every 4 training steps. When the λ_2 is set to 0, we update the T by sampling from the full CelebA training set, while when the λ_2 is set > 0, we update the T by sampling from a subset of the CelebA training set which only contains high-quality front-view faces.

3.4.2 *Iterative generation*

In the iterative generation mode, we frame the facial parameter generation as a parameter searching process. In this case, we remove the translator T and its loopback loss $\mathcal{L}_{loop}$, and directly optimize on the facial parameters x from the very input-end of the renderer. We use the gradient descent to update the parameters x so that to minimize the facial similarity loss:

$$x^\star = \arg\min_x \left(\lambda_1 \mathcal{L}_{idt} + \lambda_2 \mathcal{L}_{ctt}\right), \quad \text{s.t. } x \in [0,1]. \tag{9}$$

To help understand, we can also express the above optimization process as a facial distance minimization process over the manifold S of the game characters:

$$y^\star = \arg\min_y \ \mathcal{L}_S(y, y_r), \quad \text{s.t. } y = G(x),\ x \in [0,1]. \tag{10}$$

where we aim to find an optimal face $y^* = G(x^*)$ that minimizes the distance between y and the reference face photo y_r. Fig. 5 shows an illustration of the searching process.

TABLE 2
A detailed configuration of our Imitator G.

Layer	Component	Configuration	Output Size
Conv_1	Deconv + BN + ReLU	512×4×4 / 1	4×4
Conv_2	Deconv + BN + ReLU	512×4×4 / 2	8×8
Conv_3	Deconv + BN + ReLU	512×4×4 / 2	16×16
Conv_4	Deconv + BN + ReLU	256×4×4 / 2	32×32
Conv_5	Deconv + BN + ReLU	128×4×4 / 2	64×64
Conv_6	Deconv + BN + ReLU	64×4×4 / 2	128×128
Conv_7	Deconv + BN + ReLU	64×4×4 / 2	256×256
Conv_8	Deconv	3×4×4 / 2	512×512

A detailed optimization pipeline of the iterative generation is summarized as follows:

- **Stage I** (training). Train the imitator G, face recognition network F_{recg} and the face segmentation network F_{seg}.
- **Stage II** (inference). Initialize the facial parameter x based on the "average face". Fix G, F_{recg}, F_{seg}, and update x until reach the max-number of iterations:
  - $x \leftarrow x - \mu \frac{\partial \mathcal{L}_S}{\partial x}$ (μ: learning rate).
  - Project x_i to $[0,1]$: $x_i \leftarrow \max(0, \min(x_i, 1))$.

In Stage II, we set the max-number of iteration to 50, the learning rate μ to 10, and the decay rate to 20% per 5 iterations.

3.5 Integrating facial priors

To further improve the robustness of our method, instead of predicting facial parameters directly in the original parameter space, we learn their low-dimensional representation and making predictions in their orthogonal subspace. This can be seen as an integration of facial priors or an additional regularization on the predicted parameters. We found this operation greatly improves the stability of the generated characters.

We first run our method on the faces from CelebA [38] (front-view only) to obtain a large set of facial parameters. We then learn a whitening projection P by performing singular value decomposition on the parameter matrix. The dimension reduction can be expressed as follows:

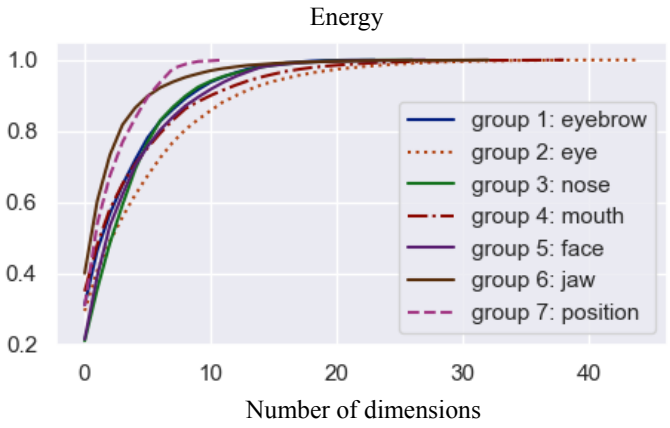
$$\hat{x} = P^T(x - m), \tag{11}$$

where m is the mean of the facial parameters which is subtracted from the input. We finally recover the predicted parameters $x^\star$ by the following inverse mapping:

$$x^\star = (PP^T)^{-1} P \hat{x}^\star + m, \tag{12}$$

where $\hat{x}^\star$ is the prediction in the low-dimensional space.

Fig. 6 (a) plots the reconstruction energy on seven groups of facial components with a different number of subspace dimensions. The curves suggest high redundancy of the original facial parameters. In Fig. 6 (b)-(c), we show the importance of the facial prior (dimension reduction) and how it affects the rendering results. We show the integration of facial priors helps generate more stable and more meaningful characters.



(a) Reconstruction energy on number of principal components

(b) Random character creation ***with facial priors***

(c) Random characters ***without facial priors***

Fig. 6. Human faces are typically embedded in a low-dimensional subspace with facial priors. We investigate the importance of the dimension reduction and how it affects the rendering results. (a) Reconstruction energy vs. Number of principal components in the parameter space of CelebA dataset. We divide the facial parameters into seven groups and perform dimension reduction accordingly. (b)-(c) Randomly generated characters w/ and w/o dimension reduction.

TABLE 3
A detailed configuration of our face segmentation model F_{seg}.

Layer	Component	Configuration	Resolution
Conv_1	Conv + BN + ReLU	64×7×7 / 2	(1/2)×(1/2)
MaxPool	MaxPool	3×3 / 2	(1/4)×(1/4)
Conv_2	3 × ResNet Block	64 / 2	(1/8)×(1/8)
Conv_3	4 × ResNet Block	128 / 1	(1/8)×(1/8)
Conv_4	6 × ResNet Block	256 / 1	(1/8)×(1/8)
Conv_5	3 × ResNet Block	512 / 1	(1/8)×(1/8)
Conv_6	Convolution	11×1×1 / 1	(1/8)×(1/8)

TABLE 4
A detailed configuration of our facial parameter translator T.

Layer	Component	Configuration
Embedding	Facial recognition network	LightCNN-29v2 [33]
FC_1	Fully-Connected	(256, 512)
Res_Att_2	Residual-Attention Block	(512, 512)
Res_Att_3	Residual-Attention Block	(512, 512)
Res_Att_4	Residual-Attention Block	(512, 512)
FC_5	Fully-Connected	(512, $n_c + n_d$)

TABLE 5
A detailed configuration of the facial customization parameters (continuous part) in the game "Justice".

Component	Controllers	# c
eyebrow-head	(t_x, t_y, t_z), $(\theta, \cancel{\phi}, \rho)$, (s_x, s_y, s_z)	8
eyebrow-body	(t_x, t_y, t_z), $(\theta, \cancel{\phi}, \rho)$, (s_x, s_y, s_z)	8
eyebrow-tail	(t_x, t_y, t_z), $(\theta, \cancel{\phi}, \rho)$, (s_x, s_y, s_z)	8
eye	(t_x, t_y, t_z), (θ, ϕ, ρ), $(\cancel{s_x}, \cancel{s_y}, \cancel{s_z})$	6
outside eyelid	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
inside eyelid	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
lower eyelid	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
inner eye corner	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
outer eye corner	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
nose body	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, $(\cancel{s_x}, \cancel{s_y}, \cancel{s_z})$	3
nose bridge	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
nose wing	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
nose tip	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
nose bottom	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
mouth	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, $(\cancel{s_x}, \cancel{s_y}, \cancel{s_z})$	3
middle upper lip	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
outer upper lip	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
middle lower lip	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
outer lower lip	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
mouth corner	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
forehead	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
glabellum	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
cheekbone	(t_x, t_y, t_z), (θ, ϕ, ρ), $(\cancel{s_x}, \cancel{s_y}, \cancel{s_z})$	5
risorius	(t_x, t_y, t_z), (θ, ϕ, ρ), $(\cancel{s_x}, \cancel{s_y}, \cancel{s_z})$	5
cheek	(t_x, t_y, t_z), (θ, ϕ, ρ), $(\cancel{s_x}, \cancel{s_y}, \cancel{s_z})$	6
jaw	$(\cancel{t_x}, t_y, t_z)$, $(\theta, \cancel{\phi}, \cancel{\rho})$, (s_x, s_y, s_z)	6
lower jaw	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
mandibular	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9
outer jaw	(t_x, t_y, t_z), (θ, ϕ, ρ), (s_x, s_y, s_z)	9

3.6 Implementation Details

Here we provide some additional implementation details on our method. We use the well-known framework PyTorch [40] to implement our method.

Imitator. Our imitator consists of eight transposed convolution layers. In each layer, the convolution kernel size is set to 4×4 and the stride of each transposed convolution layer is set to 2 so that the size of the feature map is doubled after a convolution operation. The detailed configuration of our imitator G is listed in Table 2. Specifically, in a $c \times w \times w/s$ of transposed convolution layer (denoted as "Deconv" in Table 2), c denotes the number of filters, $w \times w$ denotes the filter's size and s denotes the filter's stride. A Batch-Normalization ("BN") layer and a ReLU layer are embedded in our imitator after every convolution layers, except for the output layer. In each game, we adopt three imitators to fit three models for adult male characters, adult female characters, and young girl characters, respectively. We use the SGD optimizer to train our imitator with batch_size $= 16$, and momentum $= 0.9$. The learning rate is set to 0.01 and the learning rate decay is set to 10% per 50

Fig. 7. A visual comparison between our method and a well-known monocular 3D face reconstruction method 3DMM-CNN [2].

epochs. The training stops after 500 training epochs.

Face segmentation network. We use the Resnet-50 [15] as the backbone of our segmentation network. We remove its fully connected layers and adding a 1×1 convolution layer on its top. We also change the stride of the "Conv_3" and "Conv_4" from 2 to 1 to increase its output resolution from $1/32$ to $1/8$. The face segmentation network is first pre-trained on the ImageNet [41] and then fine-tuned on the Helen face semantic segmentation dataset [42] with 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 parameter translator. Our translator consists of a facial recognition network, three residual attention blocks and two fully connected layers. We design two individual prediction head on top of the translator network to predict continuous and discrete facial parameters separately. We finally concatenate the outputs of two heads together and feed them to our imitator or the game engine for rendering. A detailed configuration of our translator is shown in Table 4. In column "Configuration", (in_dim, out_dim)" represents the input and output dimension of its layer. "n_c" and "n_d" represent the dimensions of continuous and discrete facial parameters respectively. We follow the ResNet [15] and SENet [32], and set the layers in the residual attention blocks to "FC(512,1024) - FC(1024,512) - SE(512,16,512)".

Facial parameters. Here we use the character customization system of the game "Justice" as an example to show how the facial parameters are configured in our experiments. In "Justice", there are 264 facial parameters for "male" characters and 310 for "female" characters. Among these parameters, 208 of them are in continuous values, which are listed in Table 5. In the column "Controllers", the parameters (t_x, t_y, t_z), (θ, ϕ, ρ), and (s_x, s_y, s_z) correspond to the translation, rotation and scale changes of a facial bone on x, y and z axis respectively. The "# c" represents the number of user-adjustable controllers in each group. For those ~~strikethrough~~ controllers, their movements are banned considering the symmetry of the human face. Besides, there are additional 102 discrete parameters for female (22 for hairstyle, 36 for eyebrow style, 19 for lipstick style, and 25 for lipstick color) and 56 discrete parameters for male (23 for hairstyles, 26 for eyebrow styles, and 7 for beard styles), which are not listed in Table 5.

In our method, we encode the discrete parameters as one-hot vectors so that the prediction for the two types of parameters can be performed under a unified framework. Since it is difficult to directly optimize the one-hot vectors, we use the softmax function to smooth those binary values. The smoothing can be written as $h(x_k, \beta) = e^{\beta x_k} / \sum_{i=1}^{D'} e^{\beta x_i}, \quad k = 1, 2, \ldots, n,$

where D' represents the dimension of the discrete parameters. $\beta > 0$ controls the degree of smoothness. We set a relatively large value on β, say, $\beta = 100$, to speed up optimization in iterative mode ($\beta = 1$ in one-shot mode).

Face alignment. We perform face alignment by using the "dlib" library [43] before feeding the input face photo into our networks. We use the rendered "average face" as a reference for the alignment.

4 Experimental Results and Analysis

We test our method on two role-playing games named "Justice" and "Heaven", where the former one is a PC game launched in June 2018 with over 20 million registered users, and the latter one is a new mobile game on Android/IOS devices (coming soon). The two games are mainly developed for East Asian gamers.

We train our model on a large-scale celebrity face attributes dataset CelebA [38]. For quantitative evaluation, we test our method on multiple large scale face verification datasets, including LFW [44], CFP_FF [45], CFP_FP [45], AgeDB [46], CALFW [47], CPLFW [48], and Vggface2_FP [49]. We also build an HD celebrity dataset with 50 high-resolution facial close-up photos, all along with the images in CelebA, to conduct subjective evaluations.

4.1 Game character auto-creation

Fig. 7 shows four input face photos and their auto-customization results. We visually compares our method with a well-known monocular 3D face reconstruction method: 3DMM-CNN [2], where we can see the 3DMM-CNN only focuses on the facial outlines in its generated masks while ignores the modeling of the facial components. In Fig. 8 we give more examples of our auto-customization results. With the generated facial parameters, in-game 3D characters can be rendered by the game engine at multiple views. Fig. 9 and Fig. 10 shows some close-look, front-view customization results of our method. The generated characters share a high degree of similarity to the input photos where both of the "identity" and "expressions" are modeled although we do not make any manual adjustments on the facial parameters. Note that although the 3DMM-CNN was not initially designed for game character customization, here we still make a comparison with it in our experiment. This is because as far as we know, there are very few researches on the automatic creation of game characters and there is no open source code on this topic released yet. For more generated examples and comparison results, please refer to our supplementary material.

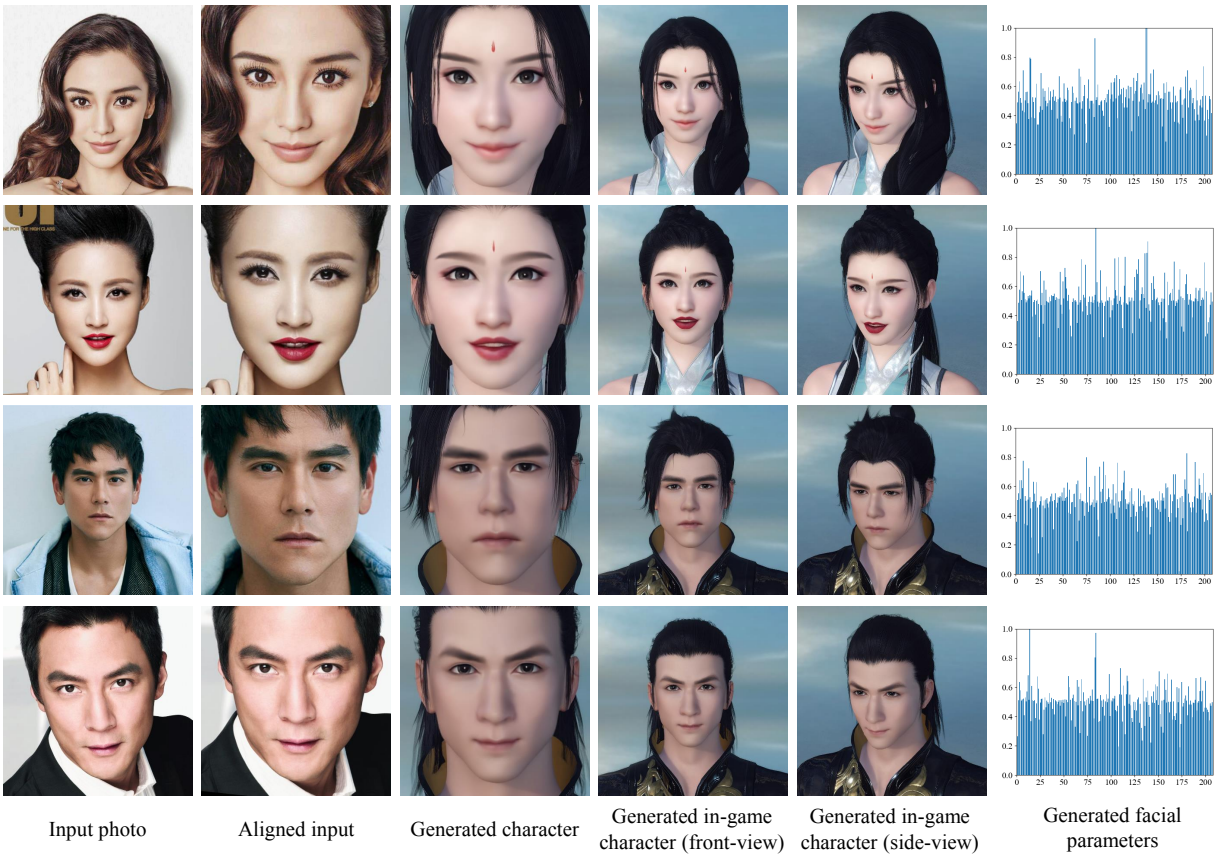


Fig. 8. Four examples of the auto-created game characters in the game "Justice" by using our method. All the example results in this figure are generated by the iterative method.

TABLE 6
Quantitative performance comparison of different methods on their accuracy and speed. The accuracy is computed on seven face verification datasets: LFW [44], CFP_FF [45], CFP_FP [45], AgeDB [46], CALFW [47], CPLFW [48], and Vggface2_FP [49]. We follow the evaluation benchmark "face.evoLVe" [50] to compute the accuracy in each of these datasets. Higher scores indicate better. Face embeddings are normalized by principal components analysis as applied in 3DMM-CNN for a fair comparison.

Method	Datasets							Speed*
	LFW	CFP_FF	CFP_FP	AgeDB	CALFW	CPLFW	Vggface2_FP	
3DMM-CNN [2]	0.9235	-	-	-	-	-	-	$\sim 10^2$Hz
Ours (iterative)	0.6977	0.7060	0.5800	0.6013	0.6547	0.6042	0.6104	$\sim$ 1Hz
Ours (one-shot)	0.9402	0.9450	0.8236	0.8408	0.8463	0.7652	0.8190	$\sim 10^3$Hz
LightCNN-29v2**	0.9958	0.9940	0.9494	0.9597	0.9433	0.8857	0.9374	$\sim 10^3$Hz

* Inference time under GTX 1080Ti. The time cost for data exchange and face alignment are not considered.
** Here we use the performance of LightCNN-29v2 on input photos as a reference (upper-bound accuracy).

TABLE 7
Subjective evaluation: selection ratio [51] of different methods on two datasets. A higher score indicates a higher user preference for the generated result.

Method	In-the-wild	HD-front view
3DMM-CNN [2]	17.4% $\pm$ 1.2%	1.7% $\pm$ 0.1%
Ours (iterative)	33.9% $\pm$ 1.2%	70.0% $\pm$ 1.1%
Ours (one-shot)	48.7% $\pm$ 1.1%	28.3% $\pm$ 1.3%

4.2 Quantitative and subjective evaluation

We also evaluate the two different versions of our method ("iterative" and "one-shot") quantitatively and subjectively on both their accuracy and speed.

To make quantitative comparisons, we follow the 3DMM-CNN [2] and use the "face verification accuracy" as our evaluation metric. We first generate facial parameters for every pair of input faces in each face verification datasets and then use the benchmarking toolkit "face.evoLVe" [50, 52, 53] to compute the verification accuracy based on the generated parameters. The intuition behind is that if the two face images belong to the same person, they should have similar facial parameters. Table 6 shows the verification scores of different methods on seven face verification datasets. A higher score suggests a better performance. The evaluation score of the 3DMM-CNN is reported by Tran *et al*. [2]. The right-side column of Table 6 shows the speed

Fig. 9. A close look at the game characters generated by our method in two games: "Justice" (first two rows) and "Heaven" (last two rows). In each group of the image, the left one shows an input photo (after alignment) and the right one shows the generated faces. All the example results in this figure are generated by the iterative method and rendered by the game engines.

performance of different methods.

The subjective comparisons are conducted on two datasets, an "HD front-view" dataset, which consists of 50 high-resolution facial close-up photos and an "in-the-wild" dataset where the face images are collected from the CelebA and are captured in an open environment with different poses, occlusions, and light conditions. We follow the subjective evaluation used by Wolf *et al*. [51] and invite 15 non-professional volunteers to rank the results generated by different methods, in which the generated characters are in random order. We define the "selection ratio" of an output character as the percentage of volunteers who select the character in each group. Finally, the overall selection ratio is used to evaluate the quality of the results generated by each method. Due to a large number of images, we only randomly select 50 images from CelebA test set for a proxy evaluation. Table 7 shows the subjective evaluation results of different methods on the above two datasets.

Together from Table 6 and Table 7 we can see that our method has higher accuracy than the 3DMM-CNN in terms of both quantitative and subjective evaluations. We also observe that the "one-shot" version of our method has a much faster speed than other approaches (1000x over the "iterative" version and 10x over the 3DMM-CNN) and also is more robust on the face images from the open environment (on the "in-the-wild" dataset). However, when dealing with the HD front-view images, the "iterative" version of our method can better catch the facial details but at the same time has a lower speed. We believe the reason why the scores in Table 6 and Table 7 are inconsistent is because that the criteria we used have different focuses. The quantitative indicator we used (face verification accuracy) pays more attention to the consistency of the facial identity, while under subjective evaluation, people tend to pay more attention to the generated facial details.

4.3 Robustness

We further test the robustness of our method on the faces with different poses, different light and blurring conditions. Fig. 11 shows some examples of the generation results.

Not limited to real photos, our method can also generate vivid game characters for some artistic portraits, including the sketch image and caricature. Although these images are either collected from an open environment or in totally different styles, we still obtain high-quality results.

4.4 Controlled experiment

Here we discuss the importance of each technical component of the proposed method and how they contribute to the result, including:

- the facial identity loss (see $\mathcal{L}_{idt}$ in Eq. 5 and Fig. 3 (b));
- the facial content loss (see $\mathcal{L}_{ctt}$ in Eq. 6 and Fig. 3 (c));
- the loopback loss (see $\mathcal{L}_{loop}$ in Eq. 7);

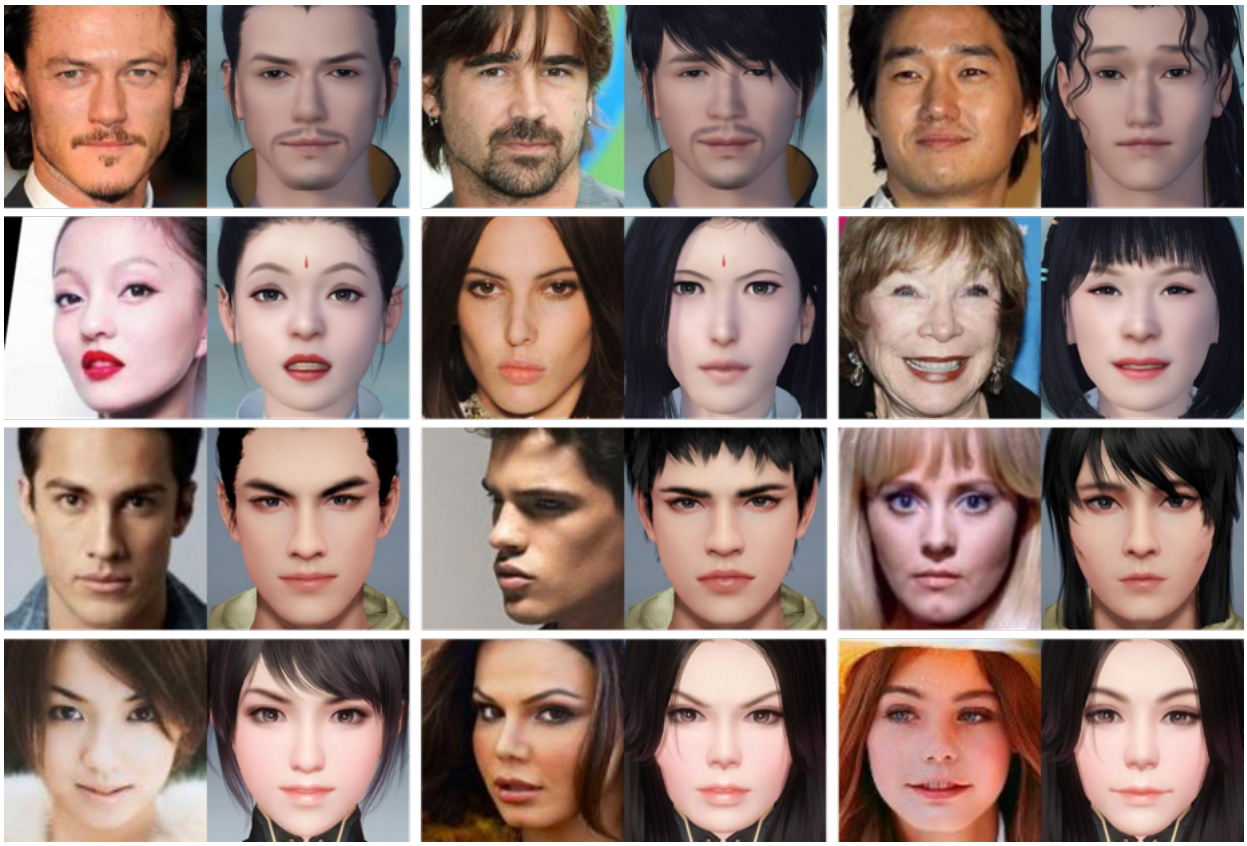


Fig. 10. A close look at the game characters generated by our method in two games: "Justice" (first two rows) and "Heaven" (last two rows). In each group of the image, the left one shows an input photo (after alignment) and the right one shows the generated faces. All the example results in this figure are generated by the one-shot method and rendered by the imitators.

TABLE 8
Ablations studies on different technical components of our method: 1) the facial content loss $\mathcal{L}_{ctt}$, 2) the facial identity loss $\mathcal{L}_{idt}$, 3) the loopback loss $\mathcal{L}_{loop}$, and 4) the residual attention block (ResAtt) in our translator. We show the integration of the above components yields consistent improvements in face verification accuracy.

Ablations				Datasets						
$\mathcal{L}_{ctt}$	$\mathcal{L}_{idt}$	$\mathcal{L}_{loop}$	ResAtt	LFW	CFP_FF	CFP_FP	AgeDB	CALFW	CPLFW	Vggface2_FP
✓	×	×	✓	0.7880	0.7930	0.6666	0.6868	0.6792	0.6252	0.6696
✓	✓	×	✓	0.8843	0.8777	0.7507	0.7917	0.7675	0.7032	0.7432
✓	✓	✓	×	0.8870	0.8901	0.7626	0.7875	0.7725	0.7042	0.7618
✓	✓	✓	✓	0.9243	0.9200	0.7896	0.8152	0.8130	0.7400	0.7854
LightCNN-29v2*				0.9948	0.9939	0.9476	0.9537	0.9438	0.8872	0.9326

* Here we use the performance of LightCNN-29v2 on input photos as a reference (upper-bound accuracy).

TABLE 9
Subjective evaluation results of two technical components of our method 1) facial identity loss $\mathcal{L}_{idt}$, and 2) facial content loss $\mathcal{L}_{ctt}$. A higher selection ration indicates a higher user preference for the result.

Ablations		
Identity loss $\mathcal{L}_{idt}$	Content loss $\mathcal{L}_{ctt}$	Selection Ratio
✓	×	$13.47\% \pm 0.38\%$
×	✓	$36.27\% \pm 0.98\%$
✓	✓	$50.26\% \pm 0.40\%$

- the residual attention mechanism in our facial parameter translator, (see Fig. 3 (d)).

Ablation studies are conducted to analyze the importance of each of the above components. We use the same evaluation metric as we used in Table 6. We first evaluate the baseline of our method, where we train our model only based on the facial content loss, then we gradually add other loss components. All evaluations are made based on the "one-shot" version of our method. To verify the effectiveness of the residual attention block in our translator, we remove all attention modules on top of the full implementation of our method and simply apply a vanilla multi-layer perceptron which is used in Genova's method [3]. Table 8 shows their performance on different face verification datasets. We observe the integration of the facial identity loss, loopback loss, and the residual attention mechanism in our translator brings consistent improvements in the accuracy. Particularly, the identity loss and the attention mechanism bring noticeable improvement to our baseline

(a) First row: input photos with pose changes and blurring. Second row: generated characters.

(b) First row: input caricature and sketch images. Second row: generated characters.

Fig. 11. (a) We test our method on faces with different poses, different light and blurring conditions. (b) Not limited to real face photos, our method can also generate realistic game characters for artistic portraits although they have completely different styles with our training data. The results in (a) are generated by the one-shot method, and the results in (b) are generated by the iterative method.

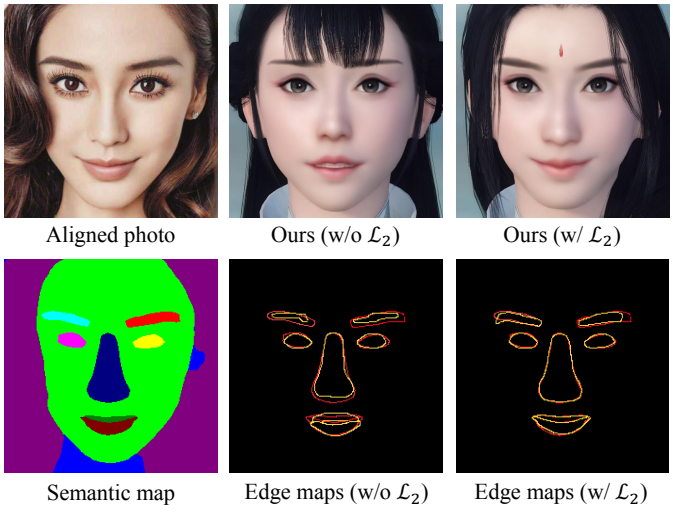


Fig. 12. (Better viewed in color) A comparison of the generated faces w/ or w/o the help of the facial content loss $\mathcal{L}_{ctt}$. The first column shows the aligned photo and its semantic map produced by our face segmentation network. The second and third columns show the generated faces w/ or w/o the help of facial content loss. Their edge maps are presented for a better visual comparison, where the yellow pixels correspond to the edge of the reference photo and the red pixels correspond to the generated faces.

method.

We also studied the visual impact on characters with different loss items. We follow the subjective evaluation method we used in Tabel 7 and analyze the user preference w/ or w/o the help of the facial identity loss $\mathcal{L}_{idt}$ and the facial content loss $\mathcal{L}_{ctt}$. Ablations are conducted on top of our full implementation. The statistics are shown in Table 9. We show that both loss items are essential for improving the visual quality of the result and the facial content loss contributes more to a higher visual preference.

Fig. 12 shows a comparison of the generated faces w/ or w/o the help of the facial content loss. For a better view, the facial semantic maps and the edges of the facial components of the generated characters are presented. In the edge maps, the yellow pixels correspond to the edge of the reference photo and the red pixels correspond to the generated faces. We observe a better correspondence of the pixel location between the input photo and the generated face when we apply the facial content loss.

4.5 Discussion

One-shot vs. Iterative. Here we give a further discussion on the the difference and connection between the two methods (one-shot vs. iterative). As we mentioned in 4.2, the two approaches have different properties in different application scenarios. We show that on HD front-view face photos, the iterative method can better catch the details than the one-shot method. On the face photos captured in the wild, the one-shot method is more robust than the iterative method. To discuss the two methods more clearly, let's consider a special case where we only have a single face image in our training set. In this case, the two methods will become almost equivalent – the only difference is that the one-shot method optimizes parameters indirectly through the T and the iterative method optimizes directly on the parameters themselves. This means that if the T has a large enough capacity and if its loss surface is smooth, the two approaches should have similar performance. However, since the face parameters are nonlinear and their dimensions are usually highly coupled, we found that the T is usually difficult to fit all training images, particularly on large-scale datasets (e.g. CelebA). In other words, it is always easy to "overfit" on a single training sample but hard on many, especially when the model capacity is limited, and the parameter space is highly complex. Therefore, in the one-shot method, sometimes we can see the facial details are not recovered very well.

In real application scenarios, we would recommend the following as an automatic way to choose a better mode for arbitrary input. Without considering the speed, we recom-

(a) (b) (c)

Fig. 13. A failure case of our iterative method. (a) shows an input face photo with pose and (b) shows an overfitted generation result of our iterative method. As we can see, although the 2D facial components are well aligned, the 3D structures are ignored. For a better comparison, in (c), we show another front-view face photo of the same celebrity shown in (a).

mend using the iterative method on HD front-view images and using the one-shot method on the faces captured in the wild. When speed is the priority, the one-shot method is clearly a better choice.

Limitation and future work. Limitation of our method is twofold. First, we found in our experiment that the iterative version of our method may fail on the faces with poses. The possible reason behind is that the iterative method tends to overfit on the 2D facial component layout and ignores the 3D structures. Fig. 13 shows a failure case of our iterative method. Our second limitation lies in our imitator. Since we use a single neural network to imitate the behavior of the renderer, it may be difficult to deal with the face models with large deformations. Fortunately, the faces in RPGs are usually built with simple deformation basis. That is to say, our methods can be easily applied to most RPG environments. In other fields, such as animated movies where character faces are typical with a more complex deformation space, we may need to design a more complex differentiable rendering model than what we used in our paper. This will be one of our future research direction (e.g., using differentiable mesh renderer or neural mesh renderer). Another of our future work is to find a potential solution to unify the two approaches into a single pipeline and complement each other.

5 Conclusion

We propose a novel method to automatically create characters in game environments based on a single input face photo. We frame the auto-creation under a self-supervised learning paradigm by leveraging the differentiable neural rendering, which bridges the gap between the deep learning and game graphics. We propose two generation modes based on this framework, namely, a one-shot generation mode and an iterative generation mode, and show in different aspects of their advantages, such as the generation quality and speed. Comparison results and ablation analysis on seven face verification benchmark datasets and a high-resolution celebrity dataset suggest the effectiveness of our method. Our method achieves a high degree of generation similarity and robustness between the input face photo and the rendered in-game character. Our method also proves to be robust to pose variants and image style changes.

Acknowledgments

The authors would like to thank the development teams of the game “Justice” and “Heaven (Mobile)”. Thanks for their valuable help and their beautiful 3D character models.

References

[1] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3d faces,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 187–194.

[2] A. Tuan Tran, T. Hassner, I. Masi, and G. 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.

[3] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman, “Unsupervised training for 3d morphable model regression,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8377–8386.

[4] B. Gecer, S. Ploumpis, I. Kotsia, and S. Zafeiriou, “Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1155–1164.

[5] T. Shi, Y. Yuan, C. Fan, Z. Zou, Z. Shi, and Y. Liu, “Face-to-parameter translation for game character auto-creation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 161–170.

[6] T. Shi, Z. Zou, Y. Yuan, and C. Fan, “Fast and robust face-to-parameter translation for game character auto-creation,” in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[7] V. Blanz and T. Vetter, “Face recognition based on fitting a 3d morphable model,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 9, pp. 1063–1074, 2003.

[8] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, “A 3d face model for pose and illumination invariant face recognition,” in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*. Ieee, 2009, pp. 296–301.

[9] O. Aldrian and W. A. Smith, “Inverse rendering of faces with a 3d morphable model,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 5, pp. 1080–1093, 2012.

[10] T. J. Cashman and A. W. Fitzgibbon, “What shape are dolphins? building 3d morphable models from 2d images,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 232–244, 2012.

[11] J. Booth, E. Antonakos, S. Ploumpis, G. Trigeorgis, Y. Panagakis, and S. Zafeiriou, “3d face morphable models ”in-the-wild”,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[12] W. Peng, Z. Feng, C. Xu, and Y. 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.

[13] L. Tran and X. Liu, "On learning 3d face morphable model from in-the-wild images," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] P. Dou, S. K. Shah, and I. 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.

[17] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos, "Large pose 3d face reconstruction from a single image via direct volumetric cnn regression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1031–1039.

[18] E. Richardson, M. Sela, R. Or-El, and R. Kimmel, "Learning detailed face reconstruction from a single image," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[19] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt, "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[20] A. Tuan Tran, T. Hassner, I. Masi, E. Paz, Y. Nirkin, and G. Medioni, "Extreme 3d face reconstruction: Seeing through occlusions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] L. Tran and X. Liu, "Nonlinear 3d face morphable model," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[23] M. M. Loper and M. J. Black, "Opendr: An approximate differentiable renderer," in *European Conference on Computer Vision*. Springer, 2014, pp. 154–169.

[24] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916.

[25] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[26] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 222.

[27] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision," in *Advances in Neural Information Processing Systems*, 2016, pp. 1696–1704.

[28] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang, "Rendernet: A deep convolutional network for differentiable rendering from 3d shapes," in *Advances in Neural Information Processing Systems*, 2018, pp. 7891–7901.

[29] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor *et al.*, "Neural scene representation and rendering," *Science*, vol. 360, no. 6394, pp. 1204–1210, 2018.

[30] A. Gruber, M. Fratarcangeli, G. Zoss, R. Cattaneo, T. Beeler, M. Gross, and D. Bradley, "Interactive sculpting of digital faces using an anatomical modeling paradigm," in *Computer Graphics Forum*, vol. 39, p. 2.

[31] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[33] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[34] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[35] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[36] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[37] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[38] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 248–255.

[42] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *European con-*

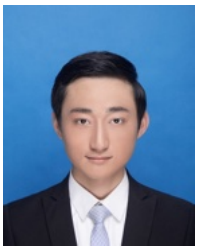
ference on computer vision. Springer, 2012, pp. 679–692.
[43] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.
[44] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008.
[45] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.
[46] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: the first manually collected, in-the-wild age database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 51–59.
[47] T. Zheng, W. Deng, and J. Hu, "Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments," *arXiv preprint arXiv:1708.08197*, 2017.
[48] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying crosspose face recognition in unconstrained environments," *Beijing University of Posts and Telecommunications, Tech. Rep*, pp. 18–01, 2018.
[49] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 67–74.
[50] J. Zhao, Y. Cheng, Y. Cheng, Y. Yang, H. Lan, F. Zhao, L. Xiong, Y. Xu, J. Li, S. Pranata *et al.*, "Look across elapse: Disentangled representation learning and photorealistic cross-age face synthesis for age-invariant face recognition," in *AAAI*, 2019.
[51] L. Wolf, Y. Taigman, and A. Polyak, "Unsupervised creation of parameterized avatars," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1530–1538.
[52] J. Zhao, J. Li, X. Tu, F. Zhao, Y. Xin, J. Xing, H. Liu, S. Yan, and J. Feng, "Multi-prototype networks for unconstrained set-based face recognition," in *IJCAI*, 2019.
[53] J. Zhao, Y. Cheng, Y. Xu, L. Xiong, J. Li, F. Zhao, K. Jayashree, S. Pranata, S. Shen, J. Xing *et al.*, "Towards pose invariant face recognition in the wild," in *CVPR*, 2018, pp. 2207–2216.

Tianyang Shi Tianyang Shi received the B.S. degree and the M.S. degree from the School of Astronautics, Beihang University in 2016 and 2019, respectively. He is currently a researcher in Netease Fuxi AI Lab. His research interests include image processing, deep learning, and their application in games.

Zhengxia Zou received his B.S. degree and his Ph.D. degree from the Image Processing Center, School of Astronautics, Beihang University in 2013 and 2018. He is now working at the Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, as a postdoc research fellow. His research interests include computer vision, pattern recognition, and remote sensing image analysis. He serves as the PC member/reviewer for several top conferences and top journals, including the NeurIPS, CVPR, AAAI, TIP, SPM, TGRS, etc. He was selected as one of the 2017 best reviewers for the *Infrared Physics and Technology*. His personal website is http://www-personal.umich.edu/~zzhengxi/.

Zhenwei Shi (M'13) received his Ph.D. degree in mathematics from Dalian University of Technology, Dalian, China, in 2005. He was a Postdoctoral Researcher in the Department of Automation, Tsinghua University, Beijing, China, from 2005 to 2007. He was Visiting Scholar in the Department of Electrical Engineering and Computer Science, Northwestern University, U.S.A., from 2013 to 2014. He is currently a professor and the dean of the Image Processing Center, School of Astronautics, Beihang University. His current research interests include remote sensing image processing and analysis, computer vision, pattern recognition, and machine learning.

Dr. Shi serves as an Associate Editor for the *Infrared Physics and Technology*. He has authored or co-authored over 100 scientific papers in refereed journals and proceedings, including the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Neural Networks, the IEEE Transactions on Geoscience and Remote Sensing, the IEEE Geoscience and Remote Sensing Letters and the IEEE Conference on Computer Vision and Pattern Recognition. His personal website is http://levir.buaa.edu.cn/.



Yi Yuan Yi Yuan received the PhD degree in Photogrammetry and Remote Sensing from Wuhan University, in 2017. He is currently a computer vision researcher in Netease Fuxi AI Lab. His research interests include face reconstruction, image generation, and transfer learning, etc.