Realistic Game Avatars Auto-Creation from Single Images via Three-pathway Network

Jiangke Lin
Netease Fuxi AI Lab
Hangzhou, China
jiangke_lin@zju.edu.cn

Lincheng Li*
Netease Fuxi AI Lab
Hangzhou, China
lilincheng@corp.netease.com

Yi Yuan
Netease Fuxi AI Lab
Hangzhou, China
yuanyi@corp.netease.com

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

***Abstract*—We propose a novel single image 3D face reconstruction method for realistic in-game avatar auto-creation. Although some existing 3D face reconstruction methods have been able to generate good geometry, there are still some shortages in texture generation, especially diffuse prediction, which limits its application in games or other scenarios. The main problems of these methods include: the details in the photo are not accurately restored, the produced diffuse is over smoothed, or the occlusion and lighting are not correctly removed, and so on. Although some methods collect high-quality 3D face data for neural networks to learn to generate realistic 3D faces, collecting 3D face data is known expensive. To address the above problems, we propose to utilize data from three sources, including single face images, manually inpainted diffuse maps paired with face portraits, and multiple photos of single IDs generated by a pretrained network. To make full use of these data, we propose a three-pathway network architecture that takes face images as input, produces diffuse maps, normal maps, as well as pose and light coefficients. The network parameters are optimized by comparing the rendered results with the input images, along with some other objective functions.**

***Index Terms*—Avatar, 3DMM, Deep Learning, 3D Face Reconstruction**

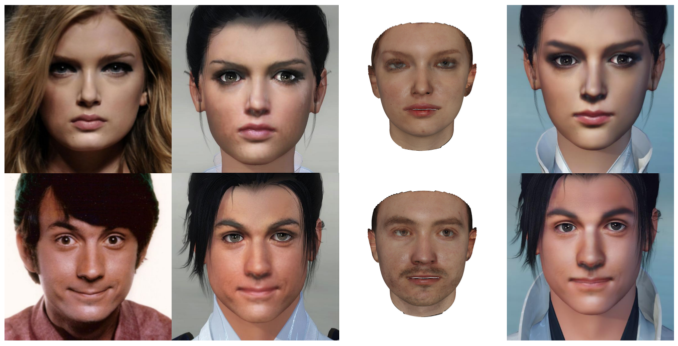

Fig. 1: From left to right: input portraits, in-game characters generated by our method and AvatarMe [5]. The avatars we generated allow players to change hairstyles and accessories at will, and can be rendered in games with different lighting and poses.

I. Introduction

Personalized game character creation is an emerging feature in recent PC and mobile games. For example, the well-known AAA game *Grand Theft Auto Online*[1] features a character creation platform where users are allowed to edit the facial appearance of their in-game characters. However, to generate desired game characters that look similar to the user themselves or their favorable celebrities, normal users would spend hours tuning hundreds of facial parameters (*e.g.* eyes, nose, and face shape) even after considerable practice.

To facilitate the process of in-game character creation, Shi *et al.* [1], [2] proposed character auto-creation methods that allow users to automatically generate corresponding face parameters from a single face photo, greatly reducing the character creation effort. However, the faces created by their method are usually limited by the degree of freedom on the facial parameters and the facial textures provided by the games. Another recent solution for creating in-game avatars is to generate a 3D head model by directly scanning a face model from multiple views with a mobile device. This technology has been adopted in some recent games such as NBA 2K[2]. However, the scan generation process would be very slow - the users usually need to wait a few minutes before finishing the creation of their characters. Besides, such an approach is not suitable for creating characters for other people such as celebrities, whose multi-view photos are hardly available for normal users.

Some recent advances in 3D face reconstruction can predicts accurate 3D faces geometry from single view images based on the 3DMM model and CNNs [3]–[9], however, the texture they produce are not sufficient for practical usage, especially in game environment. One or more defects exist in these methods: 1) the details in the photo are not accurately restored; 2) requires high-quality 3D face datasets, which are expensive to collect; 3) the produced diffuse is over smoothed; and 4) the occlusion and lighting are not correctly removed.

To generate 3D faces/heads that can be easily integrated into games with limited paired data, we propose to utilize three source of face data, including single face images, manually inpainted diffuse maps paired with face portraits, and multiple photos of single IDs generated by a pretrained network. Based on those data, a novel method for automatic game character

*Corresponding author
[1]https://www.rockstargames.com/GTAOnline
[2]https://www.nba2k.com

avatar creation from single images is also proposed in this paper. With our method, the user only needs to upload a portrait photo, and our system automatically constructs a character's realistic avatar whose face shape and appearance are similar to the input portrait in less than half a second. We take advantage of the differentiable rendering technique and proposed a rendering-in-the-loop reconstruction framework as well as a set of facial texture consistency constraints for generating realistic-looking in-game avatars. We design a three-pathway reconstruction architecture that is trained in a semi-supervised fashion with both limited labeled data (2D face images and ground truth diffuse maps) and a large number of auxiliary unlabeled data (real 2D face images and synthetic 2D face images).

Our method consists of four modules. In each training iteration, a batch of input face images are first processed by a "Shape Reconstruction module" to produce game head 3D meshes. Then the input face images are transformed to UV space by a "UV Unwrapping module". The produced coarse UV maps, as well as the input face images, are then fed to a "Texture Prediction module" and generate diffuse maps, normal maps, and the lighting coefficients for the game mesh. Finally, we introduce a "Differentiable Rendering module" in the training loop that renders the facial images based on the above predictions and enforces the rendering output to be similar to the input as much as possible. To improve the robustness of various poses and illuminations, we also introduce three discriminators and use adversarial training to improve the fidelity of the predicted facial textures. The modules in the three pathways are training jointly by using both labeled and unlabeled data.

In summary, this paper makes the following contributions:

- We introduce a rendering-in-the-loop in-game avatar creation method with self-supervised training that greatly reduces the cost of dataset acquisition while maintaining robustness and fidelity of the reconstruction.
- We propose a three-pathway reconstruction architecture, which utilizes both labeled and unlabeled data and texture consistency constraints to generate realistic-looking avatars.
- We integrate the prediction of normal maps and also an iterative rendering design to further improve the fidelity of the created 3D avatars. In particular, normal maps can greatly improve the geometry details of the low-resolution head meshes visually.

II. Related Works

Most 3D face reconstruction methods nowadays are derived from 3D Morphable Face Models (3DMM). 3DMM was first introduced by [10]. Since then, multiple variations of 3DMM have been proposed [11]–[16]. Based on the Pricipal Compoment Analysis (PCA), 3DMM produce low-dimensinal representations of the 3D face, including identity, expression and texture. The Basel Face Model (BFM) [11] is one of the most popular 3DMM variants. After scanning human faces into point clouds, it unifies them into a template mesh by the Nonrigid ICP algorithm. Then, the 3DMM is constructed from PCA and dimensionality redution.

A group of methods iteratively fit a 3DMM into the input 2D image to reconstruct the 3D face from a single image. However, ambiguities occur when the expression, pose and illumination is different from the fitted model. Much effects [17]–[19] are made to increase the robustness of 3DMM fitting, but they still suffer from complicated expression and illuminations of in-the-wild images. On the other hand, deep learning based methods directly predict the 3DMM coefficients from images as a regression problem. In order to construct paired 2D-3D data as supervision, Richardson *et al.* [20], [21] generate synthetic data by randomly sampling the morphable face model. Tran *et al.* [22] create the ground truth using an iterative optimization method to fit a large number of face images.

In addition to the single image face reconstruction, another group of methods utilizes videos or multiple images to create 3D avatars. [23] propose to utilize mobile devices to capture videos of human heads and build dynamic 3D face rigs. [24] develop algorithms to create an avatar from multiple images captured with a web camera. Although these methods can generate high-fidelity 3D faces, they require videos (or multiple photos taken in a few minutes) as input and need some assistance from users, which are not feasible for fast game character avatar creation. Besides, users cannot use such methods to create characters for their favorite celebrities.

More recently, differentiable rendering techniques are introduced to the 3D face reconstruction tasks [25]–[27]. With differentiable rendering, facial textures like UV maps can be smoothly optimized within the training loop. Some of the recent approaches utilize large-scale databases of high-quality UV maps to train a neural network for predicting facial textures. For example, Deng *et al.* [28] learn a generative model called UV-GAN to complete the self-occluded regions in the facial UV map. Chen *et al.* [29] capture 366 high-quality 3D scans from 122 different subjects, and use UNets to produce displacement maps for facial details synthesis. Then the networks are trained in a semi-supervised way using labeled 3D faces. Gecer *et al.* [30] employ the adversarial learning to train a facial position map generator, and then find the optimal latent variables through non-linear optimization. However, such datasets are not publicly available. It is laborious and expensive to capture such a dataset, which is infeasible for normal users. Additionally, with the differentiable renderer, some methods have been developed to reconstruct both the facial shape and texture from a single image in an unsupervised or weakly-supervised manner, thereby reducing reliance on expensive 3D face datasets. Deng *et al.* [3] propose a method to predict the 3DMM shape and texture coefficients at the same time with image-level and perception-level losses, leading to the state-of-the-art reconstruction results. Lin *et al.* [31] further refine the facial textures from images by applying graph convolutional networks. Despite the high-fidelity reconstruction results obtained, these 3DMM based methods aim to reconstruct the 3D shape and texture for the face region rather than the whole head, which cannot be directly used for the game character creation. On the contrary, we aim to create the whole head model with a complete texture based on the input, which can be easily applied to most 3D games.

III. Proposed Method

We propose a novel three-pathway architecture for game character avatar automatic creation. We make full use of the face data and utilize three types of source data for training, *i.e.* 1) real face images with paired human annotated diffuse map, 2) single face images, and 3) generated multiple face images from same IDs. A brief overview of our method is shown in Fig. 2. A high-level abstraction of our method can be viewed as a semi-supervised rendering process, where we enforce rendered face images to be similar to the inputs by using the predicted 3D shape and facial textures. The three pathways are trained simultaneously and their functionalities are described as follows.

1) Paired Image Pathway. We use real 2D face images and their paired ground truth diffuse maps for training. The ground truth diffuse maps are adopted from MeInGame [32], which are created by manually annotate the unwrapped image. In addition to the reconstruction of the input, in this pathway, we also enforce the predicted diffuse maps to be similar to their ground truth.
2) Single Image Pathway. In this pathway, the training is fully self-supervised. The reconstruction loss is computed between the input face images and the final rendered images.
3) Generated Image Pathway. We utilize a face image generation network called DiscoFaceGAN [33] to generate face images from different identities, lighting, expressions, and poses. In a single training iteration, we feed a batch of face images generated from the same identity but different expressions, lighting, and poses. An auxiliary loss is computed across the diffuse maps generated from the neutral front face and other faces within the batch. We use such data based on a fact that it is easier to generate a complete texture map for a neutral front face. Therefore, we treat the diffuse map generated from the neutral front face image as a pseudo ground truth in this pathway.

A. Shape Reconstruction

The shape reconstruction module takes in a 2D face image and produces a 3D game head mesh of the input. We first produce a 3DMM face by using the BFM model [11] based on the input face image. The 3DMM coefficients consists of identity coefficients $c_i \in \mathbb{R}^{80}$, expression coefficients $c_e \in \mathbb{R}^{64}$, and face pose $p \in \mathbb{R}^6$. And the shape of a 3DMM face mesh S can be represented as:

$$S = S_{mean} + c_i I_{base} + c_e E_{base}, \tag{1}$$

where S_{mean} is the mean face shape, I_{base} and E_{base} are the PCA bases of identity and expression respectively.

After a 3DMM face is generated, similar to [32], we utilize Radial Basis Function (RBF) interpolation [34] to transfer the shape of 3DMM face mesh to the game template head mesh. In this way, the 3D face, as well as the predicted textures, can be directly applied to the game environment.

B. UV Unwrapping

Since CNNs are not good at handling spatial transformations, we unwrap the input image after a 3D game head mesh is restored. To do this, we first align the game face mesh to the input face image based on the pose coefficients p predicted in the shape reconstruction phase. We then project each pixel of the UV map onto the mesh surface according to the UV coordinates of each vertex and then back-project the 3D position of each vertex to the 2D input image. For each vertex, the color (RGB) value is retrieved from the projected coordinates on the 2D input image. We finally fill in the non-skin areas with random noise. The skin region is produced by a lightweight face segmentation network from [1].

C. Texture Prediction

After the UV unwrapping stage, the created coarse UV maps are fed to the networks. We train the networks to predict the diffuse maps and normal maps based on the UV maps and the input 2D face images. Fig. 2 demonstrates the details of the proposed texture prediction networks. The network generally follows an encoder-decoder design logic. In the encoder part, we build an Image Encoder to extract latent features and predict lighting and pose coefficients from input images. We build a UV Encoder that extracts latent features from input UV maps. The latent features from the two encoders are then concatenated together, fed to a Diffuse Decoder to generate final diffuse maps, which is trained to eliminated occlusion and lighting effects.

In addition to diffuse maps, we also introduce another group of encoder-decoder to predict normal maps from input images and UV maps. Input UV maps and final diffuse maps are concatenated first, then fed to the Normal Encoder. The Normal Decoder takes the concatenated latent features from the Image Encoder and the Normal Encoder. The predicted normal map will be loaded into the game along with the diffuse map and the game head mesh.

D. Iterative Face Rendering

In each training iteration, we design an iterative rendering strategy to improve the robustness of the texture prediction module on different poses and illuminations. Fig. 3 demonstrates the proposed rendering pipeline. In the first round of rendering, we render a face image based on the predicted diffuse map, normal map, and the predicted game head mesh.

In the second round of rendering, we randomly rotate the game head mesh, and re-render a new face image. We use this image as a new input and pass it through the UV Unwrapping, Texture Prediction, and Differentiable Render modules. We also introduce an additional loss to make the face images rendered in the two stages to be similar. The iterative rendering is applied to all three pathways.

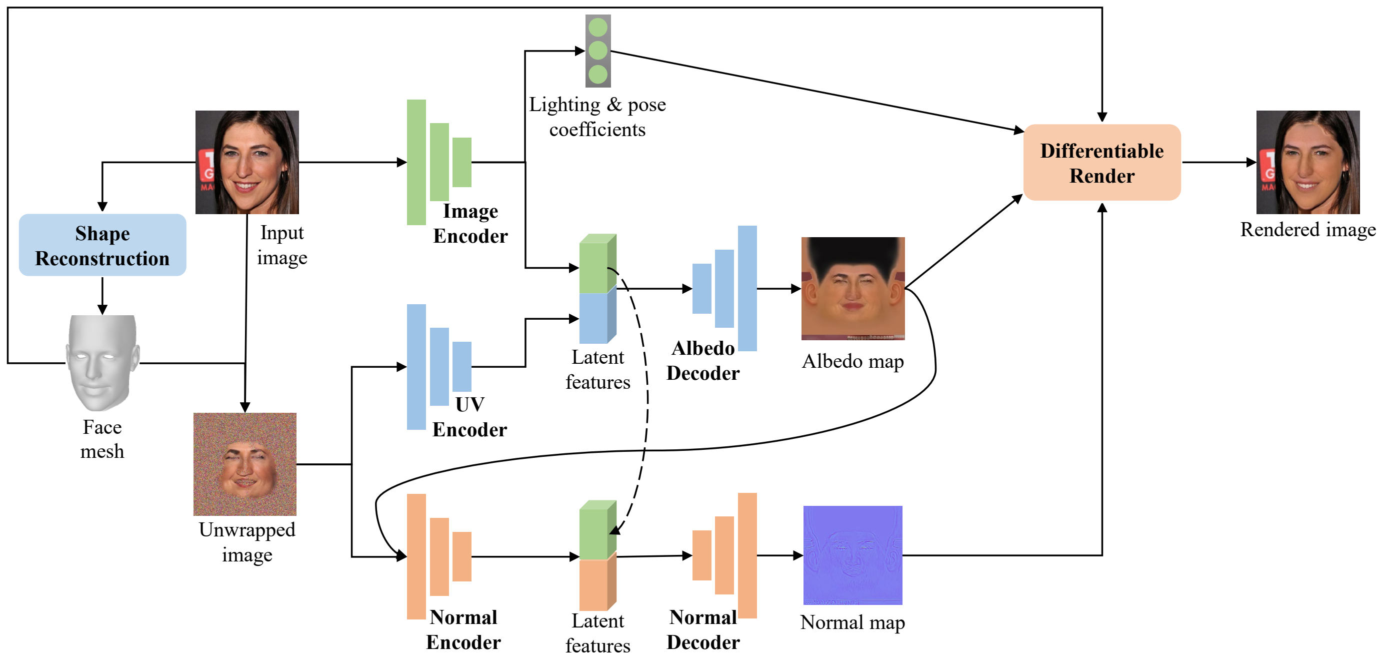

Fig. 2: A brief overview of the proposed three-pathway reconstruction architecture. Given a batch of 2D face images, we first use a shape reconstruction module to restore the corresponding 3D game head meshes. Based on the head meshes and the correspondence between vertices and UV coordinates, we unwrap the input images to UV space. A few networks then generates normal maps, diffuse maps, lighting and pose coefficients based on the input 2D images and the UV maps. A differentiable renderer is finally used to render the face images based on the predicted mesh, diffuse and normal maps.

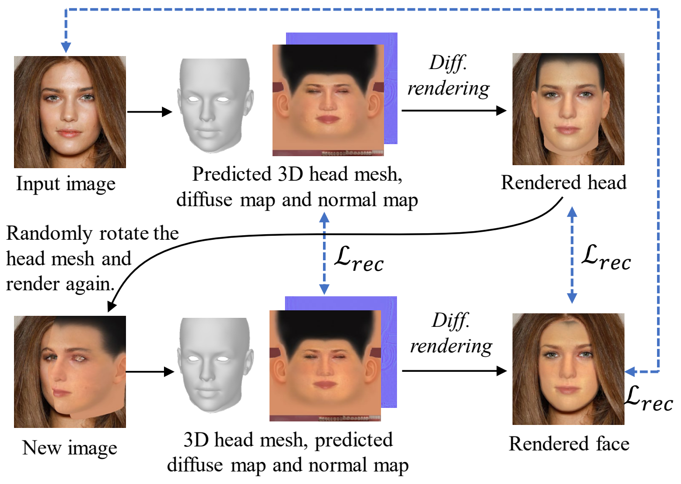
Input image
Predicted 3D head mesh, diffuse map and normal map
Diff. rendering
Rendered head
Randomly rotate the head mesh and render again.
$\mathcal{L}_{rec}$
$\mathcal{L}_{rec}$
New image
3D head mesh, predicted diffuse map and normal map
Diff. rendering
Rendered face
$\mathcal{L}_{rec}$

Fig. 3: An illustration of the proposed iterative face rendering pipeline, and the texture consistency losses.

E. Loss Functions

We design three types of loss functions for training: 1) image reconstruction losses, 2) texture consistency and regularization losses, and 3) adversarial losses. Here we introduce each of them in detail.

1) Image Reconstruction Loss: To penalize the difference between the input face image and the final rendered output, we design two loss functions, *i.e.* pixel-wise L1 loss and perceptual loss. The perceptual loss is defined by the distance between their activation maps of a pre-trained network (*e.g.* VGG-19 [35]):

$$\mathcal{L}_{perc}(x,y) = \mathbb{E}\left[\sum_i \frac{1}{N_i}\|\phi_i(x)-\phi_i(y)\|_1\right], \tag{2}$$

where ϕ_i is the activation map of the i^{th} layer of the pre-trained network. The final image reconstruction loss can be written as follows:

$$\mathcal{L}_{rec}(x,y) = \|x-y\|_1 + \mathcal{L}_{perc}(x,y), \tag{3}$$

where the x and y are 2D images. For all three pathways, the image reconstruction losses are computed between input images and rendered images.

In the Paired Image Pathway, image reconstruction losses are also applied to penalize the difference between the generated diffuse maps and the ground truth diffuse maps. In the Synthetic Image Pathway, we compute the image reconstruction losses between the generated diffuse map from the neutral front face and those from faces of random expressions, lighting, and poses. For both the Unpaired Image Pathway and the Synthetic Image Pathway, the reconstruction losses are also applied on the generated diffuse maps between the first and second round of rendering.

2) Texture Consistency and Regularization Losses: We introduce several loss functions to regularize the generated diffuse maps, making them satisfy symmetry and global consistency.

Due to the symmetry of a human face, we use a multi-scale symmetric perception (MSP) loss to ensure the symmetry of the predicted diffuse maps. The MSP loss penalizes the difference between two random horizontal symmetric patches of an image in the perception level. The MSP loss is designed at multiple image scales:

$$\mathcal{L}_{msp}(x) = \sum_{s \in S} \mathcal{L}_{perc}(\psi(x, s, r), \psi(x', s, r)), \tag{4}$$

where x' is a horizontally flipped image of x, $\psi(x, s, r)$ represents a random patch sampling function, which crops a patch of size s at random location r from image x. The S represents different scales. For an image size of 512×512 pixels, we use the scale from $\{112, 224, 336\}$ pixels.

Considering that the skin color on the entire diffuse map should be close, we minimize the standard deviation of the RGB values of the skin area on the diffuse map. However, directly minimizing the standard deviation will wrongly remove high-frequency facial details. We, therefore, calculate the standard deviation in the low-frequency space. The standard deviation loss is defined as follows:

$$\mathcal{L}_{std}(x) = \sqrt{\frac{1}{|\mathcal{M}_{skin}|} \sum_{i \in \mathcal{M}_{skin}} (\mathcal{G}(x)_i - \bar{x})^2}, \tag{5}$$

where $\mathcal{G}(x)$ represents the Gaussian Blur function, $\mathcal{M}_{skin}$ is the skin region mask, i is the pixel index, and $\bar{x}$ is the mean value of x in the skin region.

Combining the above objectives, the overall regularization loss is defined as:

$$\mathcal{L}_{dm}(x) = \mathcal{L}_{msp}(x) + \mathcal{L}_{std}(x). \tag{6}$$

Similar to the diffuse map, the normal map is also constrained by symmetry. The normal map symmetric loss is computed on a low-frequency level by L1 distance:

$$\mathcal{L}_{sym}(x) = \|\mathcal{G}(x) - \mathcal{G}(x')\|_1, \tag{7}$$

where the x' is the horizontally flipped image of input image x. To prevent the normal map from being abnormal, we penalize the distance between it and the z axis at the low-frequency level:

$$\mathcal{L}_z(x) = \|\mathcal{G}(x) - (0, 0, 1)\|_1. \tag{8}$$

The normal map regularization loss is defined as:

$$\mathcal{L}_{nm}(x) = \mathcal{L}_{sym}(x) + \mathcal{L}_z(x). \tag{9}$$

3) Adversarial losses: To further improve the fidelity of generated diffuse maps, we adopt the adversarial training method to improve the fertility of the texture prediction. Since the layout of the texture map is roughly fixed, we design three discriminators, *i.e.* a Global Discriminator, an Eye Discriminator, and a Cheek Discriminator, and placed them at fixed locations. The three discriminators are working together at global scale and two specific local regions to improve the texture maps.

The discriminators take both generated diffuse maps and ground truth diffuse maps as input, and predict whether the inputs are real or fake (generated). In the meantime, we train the rest of the networks to fool the discriminators.

The training objective is defined as follows:

$$\begin{aligned} \mathcal{L}_{gen} &= \mathbb{E}\{\log D_i(x')\} \\ \mathcal{L}_{dis} &= \mathbb{E}\{\log D_i(x)\} + \mathbb{E}\{\log(1 - D_i(x'))\}, \end{aligned} \tag{10}$$

where x is the real data and x' is the fake data. $D_i, i \in \{0, 1, 2\}$ represents the Global Discriminator, the Eye Discriminator, and the Cheek Discriminator respectively.

4) Final Loss Function: Combining all the above losses, the final loss functions of our method are defined as follows:

$$\begin{aligned} \mathcal{L}_G = {} & \lambda_1 \mathcal{L}_{rec}(d, x) + \lambda_2 \mathcal{L}_{rec}(r, m) + \lambda_3 \mathcal{L}_{dm}(d) \\ & + \lambda_4 \mathcal{L}_{nm}(n) + \lambda_5 \mathcal{L}_{gen}(d) \\ \mathcal{L}_D = {} & \lambda_5 \mathcal{L}_{dis}(d, d_{gt}), \end{aligned} \tag{11}$$

where d is the generated diffuse map, d_{gt} is the ground truth diffuse map, r is the rendered face image, m is the input image, and n is the generated normal map. x represents different variables in different pathways: In the Paired Image Pathway, x represents the ground truth diffuse maps; in the Unpaired Image Pathway, the loss term $\lambda_1 \mathcal{L}_{rec}(d, x)$ will be ignored, since there is nothing to compare with; in the Synthetic Image Pathway, x represents the generated diffuse map of the neutral front face image.

We finally train the discriminators to maximize the following objective while train the rest of the networks to minimize it:

$$G^\star, D^\star = \arg\min_G \max_D \mathcal{L}_G + \mathcal{L}_D. \tag{12}$$

Since the face renderer we used is differentiable, the training can be performed in an end-to-end fashion.

IV. Implementation Details

We use the Basel Face Model [11] for face shape reconstruction. The 3DMM face mesh contains 35,709 vertices and 70,789 faces, while the game head mesh is constructed by only 5,926 vertices and 11,482 faces. Similar to [32], we do not consider the hair, eyeballs, *etc.* in this paper. We deploy the pre-trained model provided by [3] to predict the 3DMM coefficients to reconstruct a 3DMM face mesh, but other 3DMM methods should also work.

We use the dataset created by [32] as the ground truth of our diffuse maps. The portraits from CelebA-HQ [36] are used as input face images. Due to hardware limitations, we set the resolution of both input images and texture maps to 512×512 pixels. The differentiable renderer is based on SoftRas [27], [37]. For the ground truth data, we randomly select 300 / 50 / 50 for training / evaluation / testing; for the CelebA-HQ (w/o ground truth) dataset, the numbers are 22,500 / 3,750 / 3,750; for the synthetic data, we just randomly sampled on demand.

We refer the UV Encoder, Image Encoder, Diffuse Decoder and three discriminators to "diffuse nets", and refer the Normal Encoder and Normal Decoder to "normal nets". To improve

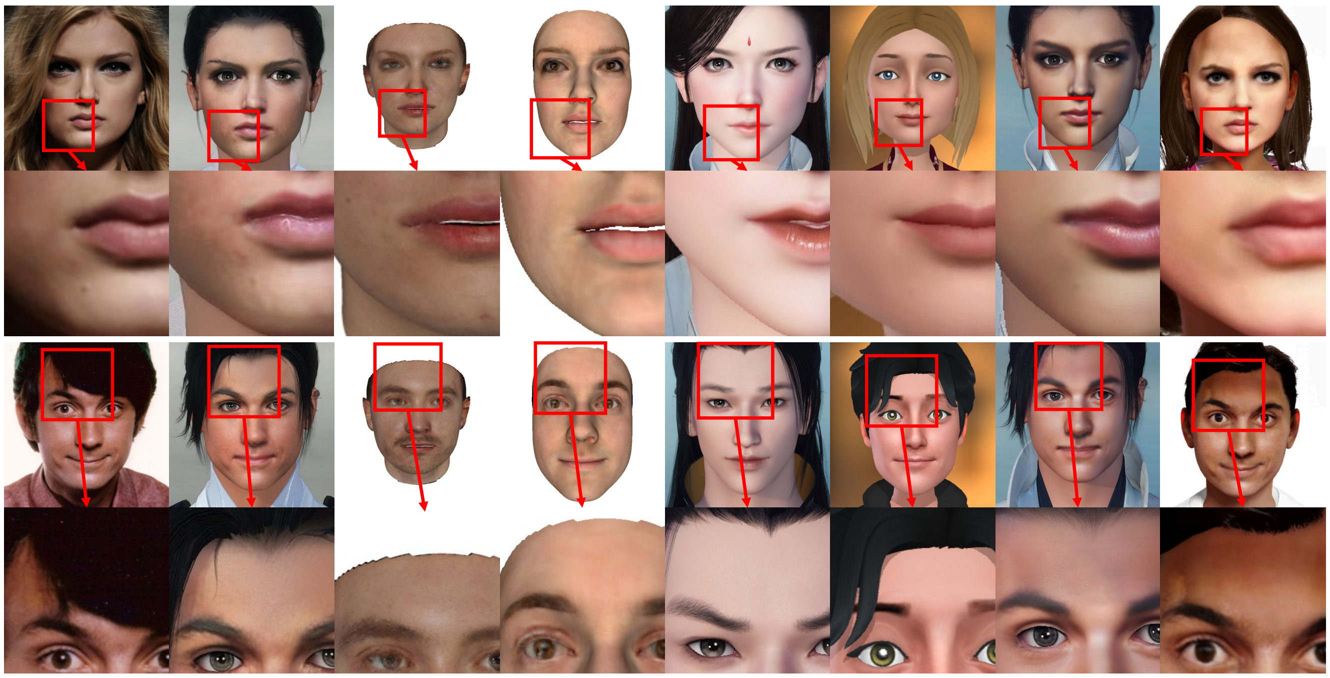

Fig. 4: Comparison of reconstructed faces with different methods used in games. The first column shows the input images. Our results are shown in the second column. The rest columns are the reconstructed 3D faces by other methods. Our results can better reconstruct occluded areas, as well as details such as moles, eyebrows, and beards. Please zoom in for more details.

the stability of training, we perform a warm-up at the beginning of the training process. The whole training is divided into four stages:

- Train the diffuse nets on the Paired Image Pathway and the Unpaired Image Pathway for 5 epochs.
- Train the diffuse nets on all three pathways for 10 epochs.
- Train the diffuse nets on all three pathways with the iterative rendering strategy for 40 epochs.
- Train the normal nets for 15 epochs.

We experimentally tested the appropriate hyper parameters and set the λs to: $\lambda_1 = 3, \lambda_2 = 3, \lambda_3 = 1, \lambda_4 = 0.1, \lambda_5 = 0.1$. The learning rate is set to 0.001 and the batch size is 4. We run our experiments on an NVIDIA 1080Ti GPU, and it normally takes less than 0.5s to generate a game avatar with the image resolution of 512×512.

V. Experimental Results

A. Qualitative Comparison

We compare our method with two state-of-the-art 3D face reconstruction methods as well as some other commercial character automatic creation systems applied in games/apps. The comparison methods include AvatarMe [5], Deep3DFaceReconstrution [3], *Justice* (https://n.163.com, which is based on Face-to-parameter (F2P) [1], [2]), *Loomie* (https://loomai.com), MeInGame [32], and *Pinscreen* (https://pinscreen.com). All these methods are designed to create 3D faces from a single image. As illustrated in Fig. 4, compared to other methods, our method can produce more realistic faces that are similar to the input images.

The faces reconstructed by *Justice* and *Loomie* have limited shape ranges and cannot faithfully represent the input face shapes. The method of *Pinscreen* can achieve similar face shapes to some extent, but it cannot handle the uneven lights, occlusions and self-occlusions very well. While AvatarMe can produce realistic 3D faces from single images, but their results cannot faithfully restore the facial details from the input images (*e.g.* the beard of the second face in Fig. 4). Compare with MeInGame, our method produces more realistic textures and can predict normal maps via self-supervised learning.

B. Quantitative Comparison

Tab. I shows the four groups of metrics computed between the input face and rendered output. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are commonly used as image quality metrics, a higher number indicates a better result.

Since it is difficult to build a set of ground truth images for game characters, another possible way for quantitative comparison is to use a pre-trained face recognition network and calculate the similarity between input images and reconstructed 3D heads [1], [2]. We adopt two pre-trained face recognition networks, LightCNN [38] and evoLVe [39], to extract feature vectors from face images. Then we calculate the cosine similarity of feature vectors between the input images and rendered face images. We compare the metrics of feature similarity extracted by pre-trained face recognition

	Ours						Deep3DFR	MeInGame
Paired	✓	✓	✓	✓	✓	✓		
Unpaired		✓	✓	✓	✓	✓		
Synthetic			✓	✓	✓	✓		
MSP Loss				✓	✓	✓		
Adv. Loss					✓	✓		
Iter. Render						✓		
PSNR ↑	26.1	26.7	26.4	26.9	**27.5**	27.3	26.6	24.8
SSIM ↑	0.81	0.84	0.82	0.84	0.87	**0.88**	0.83	0.83
LightCNN ↑	0.74	0.77	0.80	0.83	0.85	**0.87**	0.72	0.72
evoLVe ↑	0.64	0.73	0.78	0.77	0.80	**0.81**	0.64	0.62

TABLE I: Reconstruction accuracy of different methods.

AvatarMe	Face-to-Parameter	MeInGame	Ours
0.133	0.019	0.207	**0.641**

TABLE II: User study: the statistics of user preference rate on different methods.

networks between the input images and rendered 3D face meshes with [3], [32].

To further demonstrate the effectiveness of our methods, we perform a user study over the results generated by different methods. Since [32] already proved the superiority of the neural texture generation, in this experiment, we only conduct user study to compare with AvatarMe [5], MeInGame [32], and Face-to-Parameter [2]. In the user study, invite 16 non-professional volunteers and generate 39 questions for each of them. In each question, there are one input face image and screenshots of results from our method and other three methods [2], [5], [32]. Each participant was asked to choose the best result from different methods by considering both facial similarity and fidelity. Tab. II shows the proportion of the frequency of each method is selected. We can see that our user preference rate is significantly higher than other comparison methods.

C. Ablation Study

We conducted an ablation experiment to evaluate the effectiveness of different technical components of our method, including the Paired Image Pathway, Unpaired Image Pathway, Synthetic Image Pathway, multi-scale symmetric perception loss, adversarial loss, and the iterative rendering design. We gradually apply each component for training, and evaluate the results on the test set.

The results on different configuration are shown in Tab. I and Fig. 5. We aim to produce an avatar that can be loaded to games, thus, the produced diffuse maps should be reasonable for the specific game. Without using the ground truth diffuse maps, the networks may not be able to learn the distribution of real diffuse maps and generate reasonable texture, and the adversarial training cannot be performed either.

Due to the limited paired data, we deploy the Unpaired Pathway to improve the robustness and preventing over-fitting. With the Synthetic Pathway, we can guarantee the results from the same person look similar under different illumination, pose, expression, *etc.* (identity consistency). We also design a

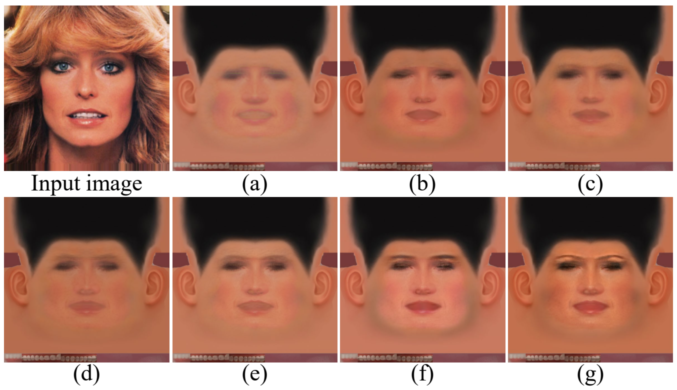

Fig. 5: Ablation study under different configurations. (a) Unpaired Image Pathway only, *i.e.* completely self-supervised, (b) Paired Image Pathway only, (c) Paired Image Pathway and Unpaired Image Pathway, (d) all three pathways, (e) three-pathway and MSP loss, (f) three-pathway, MSP loss and adversarial losses, (g) our full model.

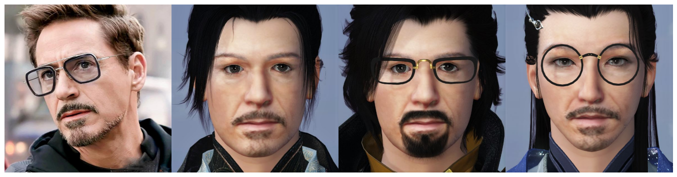

Fig. 6: Examples of user customization on top of our generated game character. Users can change hairstyles, glasses, and even facial shapes (*e.g.* the rightmost result) at will.

loss (Sec. III-E1) based on the observation that neutral front-view faces usually produce better results.

We can see as each module is applied successively, the method can obtain higher generation accuracy. Also, in Fig. 5, we can see that when the above components are added one by one, the generated facial features, especially the eyebrows, become more and more realistic.

D. Customization

After the game avatars being generated, users can optionally customize the avatars according to their preference by choosing different hairstyles, glasses, *etc*. They can also modify the facial shape of the avatars. In Fig. 6, we show a group of user customization results on top of the automatic generation result.

VI. Conclusion

We introduce a novel game character face automatic creation method, which generates realistic in-game avatars via differentiable rendering and a three-pathway generation network. The training of our method utilizes both labeled and unlabeled data, making it robust to various occlusion, lighting, and pose changes. We also design an iterative rendering strategy to improve the self-consistency of the results. Experiments and ablation studies suggest the superiority and the effectiveness of our design.

References

[1] 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/CVF International Conference on Computer Vision*, 2019, pp. 161–170.

[2] T. Shi, Z. Zuo, Y. Yuan, and C. Fan, “Fast and robust face-to-parameter translation for game character auto-creation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 1733–1740.

[3] Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong, “Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set,” in *IEEE Computer Vision and Pattern Recognition Workshops*, 2019.

[4] B. Egger, W. A. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani *et al.*, “3d morphable face models—past, present, and future,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 5, pp. 1–38, 2020.

[5] A. Lattas, S. Moschoglou, B. Gecer, S. Ploumpis, V. Triantafyllou, A. Ghosh, and S. Zafeiriou, “Avatarme: Realistically renderable 3d facial reconstruction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 760–769.

[6] Z. Zhang, Y. Ge, R. Chen, Y. Tai, Y. Yan, J. Yang, C. Wang, J. Li, and F. Huang, “Learning to aggregate and personalize 3d face from in-the-wild photo collection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 214–14 224.

[7] J. Piao, K. Sun, Q. Wang, K.-Y. Lin, and H. Li, “Inverting generative adversarial renderer for face reconstruction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 619–15 628.

[8] E. Ramon, G. Triginer, J. Escur, A. Pumarola, J. Garcia, X. Giro-i Nieto, and F. Moreno-Noguer, “H3d-net: Few-shot high-fidelity 3d head reconstruction,” *arXiv preprint arXiv:2107.12512*, 2021.

[9] A. Tewari, H.-P. Seidel, M. Elgharib, C. Theobalt *et al.*, “Learning complete 3d morphable face models from images and videos,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3361–3371.

[10] 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.

[11] 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.

[12] J. Booth, A. Roussos, A. Ponniah, D. Dunaway, and S. Zafeiriou, “Large scale 3d morphable models,” *International Journal of Computer Vision*, vol. 126, no. 2-4, pp. 233–254, 2018.

[13] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou, “Facewarehouse: A 3d facial expression database for visual computing,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 3, pp. 413–425, 2013.

[14] T. Gerig, A. Morel-Forster, C. Blumer, B. Egger, M. Luthi, S. Schönborn, and T. Vetter, “Morphable face models-an open framework,” in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 75–82.

[15] P. Huber, G. Hu, R. Tena, P. Mortazavian, P. Koppen, W. J. Christmas, M. Ratsch, and J. Kittler, “A multiresolution 3d morphable face model and fitting framework,” in *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

[16] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero, “Learning a model of facial shape and expression from 4d scans,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 194, 2017.

[17] 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.

[18] M. D. Levine and Y. C. Yu, “State-of-the-art of 3d facial reconstruction methods for face recognition based on a single 2d training image per person,” *Pattern Recognition Letters*, vol. 30, no. 10, pp. 908–913, 2009.

[19] S. Romdhani and T. Vetter, “Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 2. IEEE, 2005, pp. 986–993.

[20] E. Richardson, M. Sela, and R. Kimmel, “3d face reconstruction by learning from synthetic data,” in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 460–469.

[21] E. Richardson, M. Sela, R. Or-El, and R. Kimmel, “Learning detailed face reconstruction from a single image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1259–1268.

[22] A. Tuan Tran, T. Hassner, I. Masi, and G. Medioni, “Regressing robust and discriminative 3d morphable models with a very deep neural network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5163–5172.

[23] A. E. Ichim, S. Bouaziz, and M. Pauly, “Dynamic 3d avatar creation from hand-held video input,” *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–14, 2015.

[24] C. Cao, H. Wu, Y. Weng, T. Shao, and K. Zhou, “Real-time facial animation with image-based dynamic avatars,” *ACM Transactions on Graphics*, vol. 35, no. 4, 2016.

[25] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler, “Learning to predict 3d objects with an interpolation-based differentiable renderer,” in *Advances in Neural Information Processing Systems*, 2019, pp. 9609–9619.

[26] 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.

[27] S. Liu, T. Li, W. Chen, and H. Li, “Soft rasterizer: A differentiable renderer for image-based 3d reasoning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7708–7717.

[28] J. Deng, S. Cheng, N. Xue, Y. Zhou, and S. Zafeiriou, “Uv-gan: Adversarial facial uv map completion for pose-invariant face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7093–7102.

[29] Z. Chen, G. Zhang, Z. Zhang, K. Mitchell, J. Yu *et al.*, “Photo-realistic facial details synthesis from single immage,” *arXiv preprint arXiv:1903.10873*, 2019.

[30] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1155–1164.

[31] J. Lin, Y. Yuan, T. Shao, and K. Zhou, “Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5891–5900.

[32] J. Lin, Y. Yuan, and Z. Zou, “Meingame: Create a game character face from a single portrait,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[33] Y. Deng, J. Yang, D. Chen, F. Wen, and X. Tong, “Disentangled and controllable face image generation via 3d imitative-contrastive learning,” in *IEEE Computer Vision and Pattern Recognition*, 2020.

[34] A. De Boer, M. Van der Schoot, and H. Bijl, “Mesh deformation based on radial basis function interpolation,” *Computers & structures*, vol. 85, no. 11-14, pp. 784–795, 2007.

[35] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[36] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.

[37] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, “Accelerating 3d deep learning with pytorch3d,” *arXiv:2007.08501*, 2020.

[38] 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.

[39] 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.