MeInGame: Create a Game Character Face from a Single Portrait

Jiangke Lin, [1] **Yi Yuan,** [1*] **Zhengxia Zou** [2]
[1] Netease Fuxi AI Lab
[2] University of Michigan
[*] Corresponding Author
linjiangke@corp.netease.com, yuanyi@corp.netease.com, zzhengxi@umich.edu

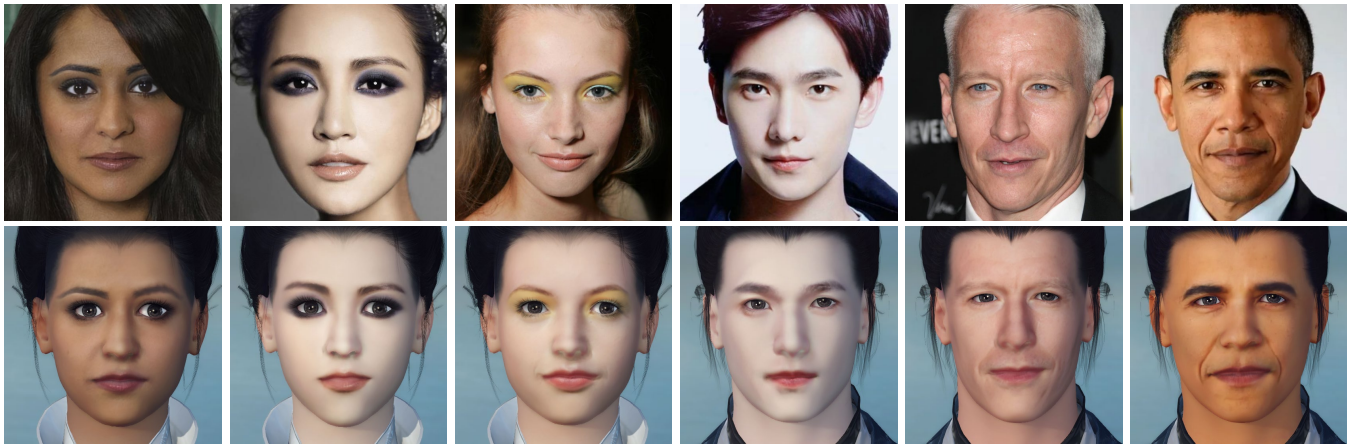

Figure 1: First row: input portraits. Second row: in-game characters generated by our method. Our method is robust to lighting changes, shadows, and occlusions, and can faithfully restore personalized details like skin tone, makeup, and wrinkles.

Abstract

Many deep learning based 3D face reconstruction methods have been proposed recently, however, few of them have applications in games. Current game character customization systems either require players to manually adjust considerable face attributes to obtain the desired face, or have limited freedom of facial shape and texture. In this paper, we propose an automatic character face creation method that predicts both facial shape and texture from a single portrait, and it can be integrated into most existing 3D games. Although 3D Morphable Face Model (3DMM) based methods can restore accurate 3D faces from single images, the topology of 3DMM mesh is different from the meshes used in most games. To acquire fidelity texture, existing methods require a large amount of face texture data for training, while building such datasets is time-consuming and laborious. Besides, such a dataset collected under laboratory conditions may not generalized well to in-the-wild situations. To tackle these problems, we propose 1) a low-cost facial texture acquisition method, 2) a shape transfer algorithm that can transform the shape of a 3DMM mesh to games, and 3) a new pipeline for training 3D game face reconstruction networks. The proposed method not only can produce detailed and vivid game characters similar to the input portrait, but can also eliminate the influence of lighting and occlusions. Experiments show that our method outperforms state-of-the-art methods used in games.

[†]Code and dataset are available at https://github.com/FuxiCV/MeInGame.

1 Introduction

Due to the COVID-19 pandemic, people have to keep social distancing. Most conferences this year have been switched to online/virtual meetings. Recently, Dr. Joshua D. Eisenberg organized a special conference, Animal Crossing Artificial Intelligence Workshop (ACAI)[1] in the Nintendo game *Animal Crossing New Horizons*[2], and has received great attention. Also, it is reported that this year's International Conference on Distributed Artificial Intelligence (DAI)[3] will also be held in the game *Justice*[4] via cloud gaming techniques. As more and more social activities come to online instead of face-to-face, in this paper, we focus on game character auto-creation, which allows users to automatically create 3D avatars in the virtual game environment by simply upload a single portrait.

Many video games feature character creation systems, which allow players to create personalized characters. How-

[1] https://acaiworkshop.com/
[2] https://www.animal-crossing.com/new-horizons/
[3] http://www.adai.ai/dai/2020/2020.html
[4] https://n.163.com/

ever, creating characters that look similar to the users themselves or their favorite celebrities is not an easy task and can be time-consuming even after considerable practices. For example, a player usually needs several hours of patience manually adjusting hundreds of parameters (e.g. face shape, eyes) to create a character similar to a specified portrait.

To improve a player's gaming experience, several approaches for game character auto-creation have emerged recently. Shi *et al*.proposed a character auto-creation method that allows users to upload single face images and automatically generate the corresponding face parameters (Shi et al. 2019). However, such a method and its latest variant (Shi et al. 2020) have limited freedom of the facial parameters and thus can not deal with buxom or slim faces very well. Besides, these methods do not take textures into account, which further limits their adaptability to different skin colors. Using a mobile device to scan a human face from multiple views to generate a 3D face model is another possible solution. The game NBA 2K[5] adopts this type of method. However, users have to wait several minutes before a character is created. Besides, this approach is not suitable for creating 3D faces for celebrities or anime characters, since their multi-view photos are hardly available for players.

To tackle the above problems, we propose a new method for automatic game character creation and a low-cost method for building a 3D face texture dataset. Given an input face photo, we first reconstruct a 3D face based on 3D Morphable Face Model (3DMM) and Convolutional Neural Networks (CNNs), then transfer the shape of the 3D face to the template game mesh. The proposed network takes in the face photo and the unwrapped coarse UV texture map as input, then predicts lighting coefficients and refined texture map. By utilizing the power of neural networks, the undesired lighting component and occlusions from the input can be effectively removed. As the rendering process of a typical game engine is not differentiable, we also take advantage of the differentiable rendering method to make gradients back-propagated from the rendering output to every module that requires parameter updating during training. In this way, all network components can be smoothly training in an end-to-end fashion. In addition to the differentiable rendering, we also design a new training pipeline based on semi-supervised learning in order to reduce the dependence of the training data. We use the paired data for supervised learning and the unlabeled data for self-supervised learning. Thus, our networks can be trained in a semi-supervised manner, reducing reliance on the pre-defined texture maps. Finally, by loading the generated face meshes and textures to the game environments, vivid in-game characters can be created for players. Various expressions can be further made on top of the created characters with blendshapes.

Our contributions are summarized as follows:

- We propose a low-cost method for 3D face dataset creation. The dataset we created is balanced in race-and-gender, with both facial shape and texture created from in-the-wild images. We will make it publicly available after the paper got accepted.
- We propose a method to transfer the reconstructed 3DMM face shape to the game mesh which can be directly used in the game environment. The proposed method is independent of the mesh connectivity and is computationally efficient for practical usage.
- To eliminate the influence of lighting and occlusions, we train a neural network to predict an integral diffuse map from a single in-the-wild human face image under an adversarial training paradigm.

[5]https://www.nba2k.com/

2 Related Work

2.1 3D Morphable Face Models

Recovering 3D information from a single 2D image has long been a challenging but important task in computer vision. 3D Morphable Face Models (3DMM), as a group of representative methods for 3D face reconstruction, was originally proposed by Blanz and Vetter (Blanz and Vetter 1999). In 3DMM and its recent variants (Booth et al. 2016; Cao et al. 2013; Gerig et al. 2018; Huber et al. 2016; Li et al. 2017), the facial identity, expression and texture are approximated by low-dimensional representations from multiple face scans.

In a typical 3DMM model, given a set of facial identity coefficients c_i and expression coefficients c_e, the face shape S can be represented as follows:

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

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

2.2 Facial Texture

Face texture restoration aims to extract textures from input face photos. A popular solution for face texture restoration is to frame this process as an image-to-texture prediction based on supervised training. Zafeiriou *et al*.captured a large scale 3D face dataset (Booth et al. 2016). They made the shape model public available but remain the texture information private. With such a private large scale dataset, Zafeiriou and his colleagues (Deng et al. 2018; Zhou et al. 2019; Gecer et al. 2019) produced good results on shape and texture restoration.

However, many 3D face reconstruction methods do not involve the face texture restoration so far since it is expensive to capture face textures. On one hand, the data acquired in controlled environments can not be easily applied to in-the-wild situations. On the other hand, it is not easy to balance subjects from different races, which may lead to bias and lack of diversity in the dataset. In particular, most subjects in (Booth et al. 2016) are Caucasian, therefore, methods based on such a dataset may not be well generalized to Asians or Africans or other races. Recently, Yang *et al*. (Yang et al. 2020) spent six months collecting a face dataset named FaceScape from 938 people (mostly Chinese) and made this dataset publicly available. However, compare to (Booth et al. 2016), FaceScape still has very limited scale.

2.3 Differentiable Rendering

To address the dataset issue, differentiable rendering techniques (Genova et al. 2018; Chen et al. 2019; Liu et al. 2019)

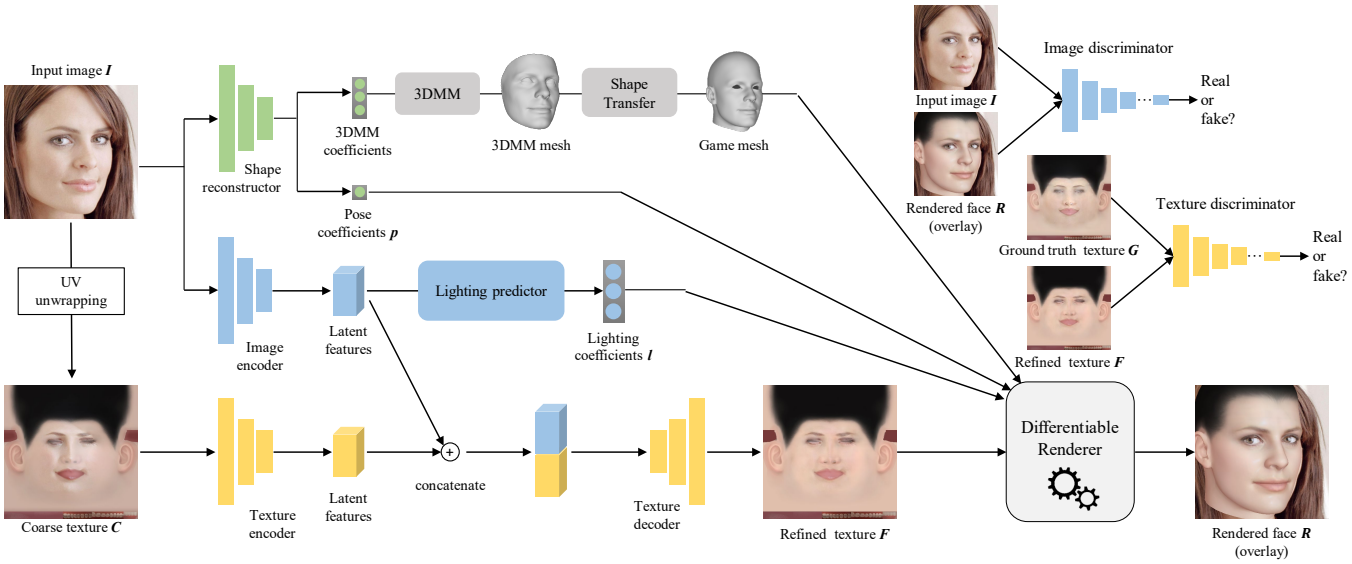

Figure 2: An overview of our method. Given an input photo, a pre-trained shape reconstructor predicts the 3DMM and pose coefficients and a shape transfer module transforms the 3DMM shape to the game mesh while keeping their topology. Then, a coarse texture map is created by unwrapping the input image to UV space based on the game mesh. The texture is further refined by a set of encoder and decoder modules. We also introduce a lighting regressor to predict lighting coefficients from image features. Finally, the predicted shape, texture, together with the lighting coefficients, are fed to a differentiable renderer, and we force the rendered output similar to the input photo. Two discriminators are introduced to further improve the results.

have been introduced to face reconstruction recently. With a differentiable renderer, an unsupervised/self-supervised loop can be designed where the predicted 3D objects (representing by meshes, point clouds, or voxels) can be effectively back-projected to 2D space, and thereby maximizes their similarity between the projection and the input image. However, even with a differentiable renderer, there is still a lack of constraints on the reconstructed 3D face information. Methods that used differentiable rendering (Genova et al. 2018; Deng et al. 2019) still rely on the prior knowledge of 3DMM, which are not fully unsupervised. Besides, such textures restored by 3DMM based methods cannot faithfully represent the personalized characteristics (e.g. makeup, moles) of the input portrait. Lin *et al.* (Lin et al. 2020) recently propose to refine the textures from images by applying graph convolutional networks. While achieving high-fidelity results, 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. As a comparison, we aim to create the whole head model with a complete texture, whose shape and appearance are similar to the input.

2.4 Shape Transfer

Shape transfer aims to transfer shapes between two meshes. To generate a full head model instead of a front face only, we use shape transfer to transfer a 3DMM mesh to a head mesh with game topology. Non-rigid Iterative Closest Point (Non-rigid ICP) algorithm (Amberg, Romdhani, and Vetter 2007) is the typical method for a shape transfer task, which performs iterative non-rigid registration between the surfaces of two meshes. Usually, non-rigid ICP and its variants has good performance on meshes with regular topology. However, such methods normally take several seconds to complete a transfer, which is not fast enough for our task.

3 Approach

Fig. 2 shows an overview of the proposed method. We frame the reconstruction of the face shape and texture as a self-supervised facial similarity measurement problem. With the help of differentiable rendering, we design a rendering loop and force the 2D face rendering from the predicted shape and texture similar to the input face photo.

Our method consists of several trainable sub-networks. The image encoder takes in the face image as input and generates latent features. The image features are then flattened and fed to the lighting regressor - a lightweight network consists of several fully-connected layers and predicts lighting coefficients (light direction, ambient, diffuse and specular color). Similar to image encoder, we introduce a texture encoder. The features of the input image and coarse texture map are concatenated together and then fed into the texture decoder, producing the refined texture map. With the game mesh, the refined texture map, pose, and lighting coefficients, we use the differentiable renderer (Ravi et al. 2020) to render the face mesh to a 2D image and enforce this image to be similar with the input face photo. To further improve the results, we also introduce two discriminators, one for the rendered face image and another for the generated face texture maps.

3.1 Dataset Creation

Here we introduce our low-cost 3D face dataset creation method. Unlike other methods that require multi-view images of subjects, which are difficult to capture, our method only uses single view images and is easily acquired. With such a method, we, therefore, create a Race-and-Gender-Balance (RGB) dataset and name it "RGB 3D face dataset".

The dataset creation includes the following steps:

i. Given an input face image, detect the skin region by using a pre-trained face segmentation network.

ii. Compute the mean color of the input face skin and transfer the mean skin color to the template texture map (provided by the game developer).

iii. Unwrapping the input face image to UV space according to the deformed game head mesh.

iv. Blend the unwrapped image with the template texture map using Poisson blending. Remove the non-skin regions such as hair and glasses, and use symmetry to patch up the occluded regions when possible.

Fig. 3 shows some texture maps created by using the above method. The texture maps with good quality are chosen for further refinement. We manually edit the coarse texture map by using an image editing tool (e.g. *PhotoShop*) to fix the shadows and highlights. Since we can control the quality of the generated texture, the workload of manual repair is very small, and each face only takes a few minutes to complete the refinement.

Figure 3: some examples of generated texture maps. the first row: input images; second row: coarse texture maps. we select those good quality texture maps (the three on the right) to further create ground truth, as shown in the last row.

3.2 Face Shape Reconstruction

The very first step of our method is to predict the 3DMM shape and pose coefficients from an input image. In this paper, We adopt the 3DMM coefficient regressor in (Deng et al. 2019) for face shape reconstruction, but other 3D face reconstruction methods will also work. Given a 2D image, we aim to predict a 257-dimensional vector $(c_i, c_e, c_t, p, l) \in \mathbb{R}^{257}$, where $c_i \in \mathbb{R}^{80}$, $c_e \in \mathbb{R}^{64}$ and $c_t \in \mathbb{R}^{80}$ represent the 3DMM identity, expression and texture coefficients respectively. $p \in \mathbb{R}^6$ is the face pose and $l \in \mathbb{R}^{27}$ represents the lightings. With the predicted coefficients, the face vertices' 3D positions S can be computed based on Eq. 1.

3.3 Shape Transfer

The shape transfer module aims to transfer the reconstructed 3DMM mesh to the game mesh. We design our shape transfer module based on Radial Basis Function (RBF) interpolation (De Boer, Van der Schoot, and Bijl 2007).

Specifically, RBF defines a series of basis functions as follows:

$$\Phi_i(x) = \varphi_i(\|x - x'\|), \tag{2}$$

where x' represents the center of φ and $\|x - x'\|$ denotes the Euclidean distance between the input point x and center x'.

In this way, given a point x and a set of RBF basis functions, the value of $f(x)$ can be computed using RBF interpolation:

$$f(x) = \sum_{i=0}^{n} w_i \varphi(\|x - x'_i\|) \tag{3}$$

where w_i represents the weight of each basis, which are the unknowns to be solved.

In the scenario of mesh shape transfer, we first manually specified 68 landmark pairs between the 3DMM template mesh and the game template mash, similar to those in dlib (King 2009). In the following, we donote those landmarks on template mesh as original face landmarks.

Then we set the centers of the basis functions x' as the original face landmarks $\widehat{L_g}$ on the game mesh. The input x is set to the original position of a game mesh vertex and the value $f(x)$ we get is the offset of the transfer. In this way, the vertex's new position can be computed as $x + f(x)$. To determine the weights w_i, we propose to solve a linear least-squares problem by minimizing the distance of the face landmark pairs between the game mesh and 3DMM mesh. For more details about the shape transfer, please refer to our supplementary materials and code.

3.4 Loss Functions

We design our loss functions to minimize the distance between the rendered face image and the input face photo, and the distance between the refined texture map and the ground truth texture map. Within the rendering loop, we design four types of loss functions, i.e. the pixel loss, the perceptual loss, the skin regularization loss, and the adversarial loss, to measure the facial similarity from both global appearance and local details.

Pixel Loss We compute our pixel loss on both of the rendered image space and the texture UV space.

For the rendered image R, the loss is computed between R and its corresponding input image I. We define the loss as the pixel-wise L1 distance between the two images:

$$\mathcal{L}_{rec}(I, R) = \frac{\sum_{i \in \mathcal{M}} \|I_i - R_i\|_1}{|\mathcal{M}_{2d}|} \tag{4}$$

where i is the pixel index, $\mathcal{M}_{2d}$ is the skin region mask obtained by the face segmentation network in 2D image space.

For the pixel loss on UV space, we define this loss as the L1 distance between the refined texture map F and the ground truth texture map G:

$$\mathcal{L}_{tex}(F, G) = \frac{1}{N} \sum_{i=0}^{N} \|F_i - G_i\|_1 \tag{5}$$

where i is the pixel index and N is the number of pixels.

Perceptual Loss We following Nazeri *et al*. (Nazeri et al. 2019) and design two losses in perception level, i.e. perceptual loss $\mathcal{L}_{perc}$ and style loss $\mathcal{L}_{style}$. Given a pair of images x and x', the perceptual loss is defined by the distance between their activation maps of a pre-trained network (e.g., VGG-19):

$$\mathcal{L}_{perc}(x, x') = \mathbb{E}\left[\sum_i \frac{1}{N_i}\|\phi_i(x) - \phi_i(x')\|_1\right] \quad (6)$$

where ϕ_i is the activation map of the i^{th} layer of the network. The style loss is on the other hand defined on the covariance matrices of the activation maps:

$$\mathcal{L}_{sty}(x, x') = \mathbb{E}_j\left[\|G_j^{\phi}(x) - G_j^{\phi}(x')\|_1\right] \quad (7)$$

where $G_j^{\phi}(x)$ is a Gram matrix constructed from activation maps ϕ_j.

We compute the above two losses above on both the face images and texture maps.

Skin Regularization Loss To produce a constant skin tone across the whole face and remove highlights and shadows, we conduct two losses to regularize the face skin, namely a "symmetric loss" and a "standard-deviation loss". Unlike previous works (Tewari et al. 2018; Deng et al. 2019) that apply skin regularization directly on vertex color, we impose the penalties on the Gaussian blurred texture map. This is based on a fact that some personalized details (e.g. a mole) are not always symmetric and not related to skin tone.

We define the symmetric loss as follows:

$$\mathcal{L}_{sym}(F') = \frac{2}{N_U \times N_V}\sum_i^{N_V}\sum_j^{N_U/2} F'_{i,j} - F'_{i,N_U-j} \quad (8)$$

where F' is the Gaussian blurred refined texture map F. N_U and N_V are the numbers of columns and rows of the texture map respectively.

We define the skin standard-deviation loss as follows:

$$\mathcal{L}_{std}(F') = \sqrt{\frac{1}{|\mathcal{M}_{uv}|}\sum_{i\in\mathcal{M}_{uv}}(F'_i - \bar{F}')^2} \quad (9)$$

where the $\bar{F}'$ is the mean value of F', $\mathcal{M}_{uv}$ is the skin region mask in UV space and i is the pixel index.

Adversarial Loss To further improve the fidelity of the reconstruction, we also use adversarial losses during the training. We introduce two discriminators, one for the rendered face and one for the generated UV texture maps, respectively. We train the discriminators to tell whether the generated outputs are real or fake, at the same time, we train other parts of our networks to fool the discriminators. The objective functions of the adversarial training are defined are follows:

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

where $D_i \in \{D_{img}, D_{tex}\}$ are the discriminators for image and texture map separately.

Final Loss Function By combining all the above defined losses, our final loss functions can be written as follows:

$$\begin{aligned}\mathcal{L}_G =& \lambda_{l1}(\mathcal{L}_{ren}(I, R) + \mathcal{L}_{tex}(F, G))\\ &+ \lambda_{perc}(\mathcal{L}_{perc}(I, R) + \mathcal{L}_{perc}(F, G))\\ &+ \lambda_{sty}(\mathcal{L}_{sty}(I, R) + \mathcal{L}_{sty}(F, G))\\ &+ \lambda_{sym}\mathcal{L}_{sym}(F') + \lambda_{std}\mathcal{L}_{std}(F')\\ &+ \lambda_{adv}(\mathcal{L}_{gen}(R|D_{img}) + \mathcal{L}_{gen}(F|D_{tex})),\end{aligned} \quad (11)$$

$$\mathcal{L}_D = \lambda_{adv}(\mathcal{L}_{dis}(I, R|D_{img}) + \mathcal{L}_{dis}(F, G|D_{tex})), \quad (12)$$

where $\mathcal{L}_G$ is the loss for training the image encoder, texture map encoder, light regressor and texture map decoder. $\mathcal{L}_D$ is the loss for training the discriminators. λs are the corresponding weights to balance the different loss terms.

During the training, we aim to solve the following minimax optimization problem: $\min_G \max_D \mathcal{L}_G + \mathcal{L}_D$. In this way, all the network components to be optimized can be trained in an end-to-end fashion. For these texture maps that do not have paired ground truth data, we simply ignore corresponding loss items during the training process.

4 Implementation Details

We use the Basel Face Model (Paysan et al. 2009) as our 3DMM. We adopt the pre-trained network from (Deng et al. 2019) to predict the 3DMM and pose coefficients. The 3DMM face we used contains 35,709 vertices and 70,789 faces. We employ the head mesh from a real game, which contains 8,520 vertices and 16,020 faces. The facial segmentation network we used is from (Shi et al. 2019).

We use the CelebA-HQ dataset (Karras et al. 2017) to create our dataset. During the UV unwrapping process, we set the resolution of the face images and the unwrapped texture maps to 512×512 and $1,024 \times 1,024$ respectively. The dataset we created consists of six subsets: {Caucasian, Asian and African} $\times$ {female and male}. Each subset contains 400 texture maps. From each subset, we randomly select 300 for training, 50 for evaluation, and 50 for testing.

Note that the game mesh we produced only corresponds to an enclosed head. Hair, beard, eyeballs, etc. are not considered in our method. This is because, in many games, heads, hair, etc. are topologically independent modules. Players can freely change various hairstyles (including long hair, etc.) and other decorations without affecting the topological structure of the head.

We use grid-search on the loss weights from 0.001 to 10, and we select the best configuration based on the overall loss on the validation set as well as quantitative comparison. The weights of loss terms are finally set as follows: $\lambda_{l1} = 3, \lambda_{perc} = 1$, $\lambda_{sty} = 1$, $\lambda_{sym} = 0.1$, $\lambda_{std} = 3$, $\lambda_{adv} = 0.001$. The learning rate is set to 0.0001, we use the Adam optimizer and train our networks for 50 epochs.

We run our experiments on an Intel i7 CPU and an NVIDIA 1080Ti GPU, with PyTorch3D (v0.2.0) and its dependencies. Given a portrait and coarse texture map, our network only takes 0.4s to produce a $1,024 \times 1,024$ refined texture map.

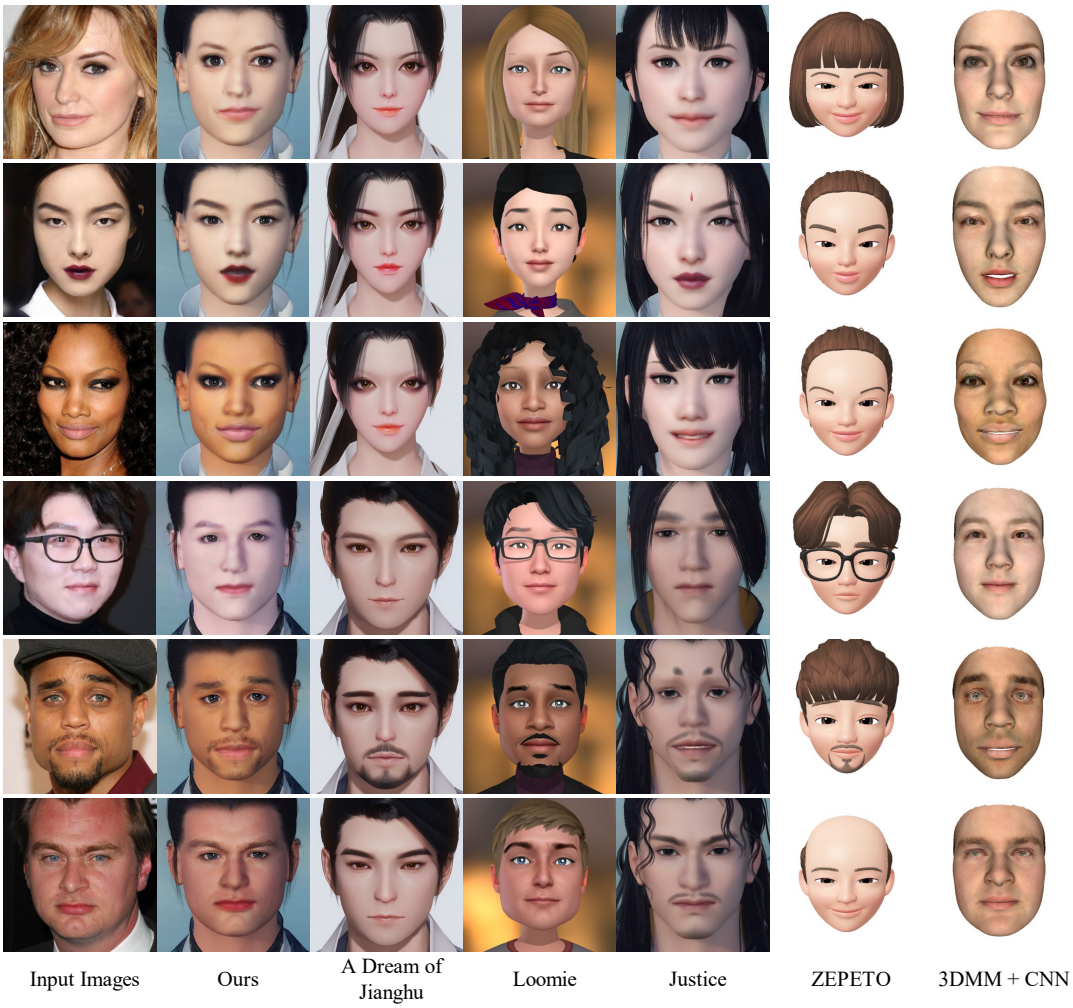

Figure 4: Comparison with other methods used in games: *A Dream of Jianghu*, *Loomie*, *Justice* (Shi et al. 2020), *ZEPETO*. We also show the results of a 3DMM-based method (Deng et al. 2019) in the last column as references.

5 Experimental Results

5.1 Qualitative Comparison

We compare our method with some other state-of-the-art game character auto-creation methods/systems, including the character customization systems in *A Dream of Jianghu*[6], *Loomie*[7], *Justice*[8] (which is based on the method of (Shi et al. 2020)), and *ZEPETO*[9].

As shown in Fig. 4, our results are more similar to the input images than the other results in both of the face shape and appearance. The faces reconstructed by *Justice* (Shi et al. 2020), *A Dream of Jianghu*, and *ZEPETO* have limited shape variance, and also fail to recover the textures of the input images. *Loomie* restores both facial shape and texture, but it cannot handle difficult lighting conditions (e.g. highlights), occluded regions (e.g. eyebrow of the first example), and personalized details (e.g. makeup).

We also compare with the state-of-the-art 3DMM based method (Deng et al. 2019) in Fig. 4 which applies CNNs to reconstruct 3DMM faces from single images. We can see that the 3DMM only models facial features and does not include a complete head model as well as textures, making it difficult to be directly used in the game environments.

For more comparisons please refer to our supplementary materials.

5.2 Quantitative Comparison

Since it is hard to acquire ground truth data of game characters, we perform a user study between our results and others. We invited 30 people to conduct the evaluation. Each person was assigned with 480 groups of results. Each group of results included a portrait, our result, and a result from others. Participants were asked to choose a better one from the two results by comparing them with the reference portrait. We believe the user reported score reflects the quality of the results more faithfully than other indirect metrics. The statis-

[6]https://jianghu.163.com
[7]https://loomai.com
[8]https://n.163.com
[9]https://zepeto.me

	A Dream of Jianghu	Loomie	Justice	ZEPETO
Others / Ours	0.0075 / 0.9925	0.0394 / 0.9606	0.0336 / 0.9644	0.0081 / 0.9919

Table 1: User preferences of the character auto-creation methods featured in different games.

	PSNR	SSIM
WildUV (Deng et al. 2018)	22.9	0.887
RGB 3D Face (ours)	24.2	0.905

Table 2: PSNR and SSIM of our method and Deng's method on RGB 3D dataset and WildUV dataset.

tics of the user study results are shown in Tab. 1. The user study shows that ours are significantly better than others.

In addition to the user study on the final reconstructions, we also compare our method with Deng *et al*. (Deng et al. 2018) on the quality of restored textures. Deng *et al*.did not take lighting or occlusions into consideration, which makes their method more like image inpainting than texture reconstruction. We compute Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) metric between the refined texture maps and the ground truth texture maps. The scores are shown in Tab. 2. Note that Deng *et al*. (Deng et al. 2018) reported their results on WildUV, a dataset similar to ours which is also constructed from an in-the-wild dataset (UMD video dataset (Bansal et al. 2017)). A direct comparison with our results on RGB 3D Face could be unfair to some extent. Nevertheless, here we still list their result as a reference.

5.3 Ablation Study

To evaluate the contribution of different loss functions, we conduct ablation studies on perceptual loss, skin regularization loss, and adversarial loss. Some examples are demonstrated in Fig. 5, the full model generates more realistic texture maps.

Figure 5: Ablation study on different loss functions.

We also compute the PSNR and SSIM metrics, the scores are shown in Tab. 3.

Losses	Pixel	✓	✓	✓	✓
	Perceptual		✓	✓	✓
	Skin Regularization	✓		✓	✓
	Adversarial	✓	✓		✓
PSNR ↑		24.10	23.83	24.13	**24.21**
SSIM ↑		0.901	0.898	0.904	**0.905**

Table 3: Metrics on ablation study.

5.4 Discussion

Although we achieve higher accuracy than other methods in quantitative and qualitative metrics, our method still has some limitations. As shown in Fig. 6 (a), when there are heavy occlusions (e.g., the hat), our method fails to produce faithful results since the renderer fails to model the shadow created by the objects outside the head mesh.

Fig. 6 (b, c) show the results from two portraits of the same person under severe lighting changes. Given (b) or (c) alone, either of the results looks good. Theoretically, it should produce similar results for the same person. However, the results are affected by lights of different colors.

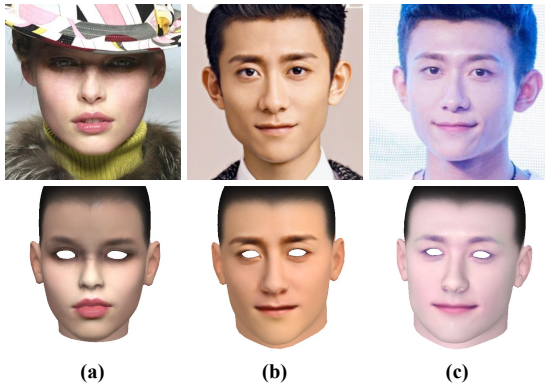

Figure 6: Special cases of the proposed method, a) shadows caused by hats that cannot be modeled by the differentiable renderer, b) and c) show the same person and the result under severe lighting changes.

6 Conclusion

In this paper, we present a novel method for automatic creation of game character faces. Our method produces character faces similar to the input photo in terms of both face shape and textures. Considering it is expensive to build 3D face datasets with both shape and texture, we propose a low-cost alternative to generate the data we need for training. We introduce a neural network that takes in a face image and a coarse texture map as inputs and predicts a refined texture map as well as lighting coefficients. The highlights, shadows, and occlusions are removed from the refined texture map and personalized details are preserved. We evaluate our method quantitatively and qualitatively. Experiments demonstrate our method outperforms the existing methods applied in games.

References

Amberg, B.; Romdhani, S.; and Vetter, T. 2007. Optimal step nonrigid ICP algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. IEEE.

Bansal, A.; Castillo, C.; Ranjan, R.; and Chellappa, R. 2017. The do's and don'ts for cnn-based face verification. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2545–2554.

Blanz, V.; and Vetter, T. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 187–194.

Booth, J.; Roussos, A.; Zafeiriou, S.; Ponniah, A.; and Dunaway, D. 2016. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5543–5552.

Cao, C.; Weng, Y.; Zhou, S.; Tong, Y.; and Zhou, K. 2013. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics* 20(3): 413–425.

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

De Boer, A.; Van der Schoot, M.; and Bijl, H. 2007. Mesh deformation based on radial basis function interpolation. *Computers & structures* 85(11-14): 784–795.

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

Deng, Y.; Yang, J.; Xu, S.; Chen, D.; Jia, Y.; and Tong, X. 2019. Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set. In *IEEE Computer Vision and Pattern Recognition Workshops*.

Gecer, B.; Ploumpis, S.; Kotsia, I.; and Zafeiriou, S. 2019. GANFIT: Generative adversarial network fitting for high fidelity 3D face reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1155–1164.

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

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

Huber, P.; Hu, G.; Tena, R.; Mortazavian, P.; Koppen, P.; Christmas, W. J.; Ratsch, M.; and Kittler, J. 2016. 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*.

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

King, D. E. 2009. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research* 10: 1755–1758.

Li, T.; Bolkart, T.; Black, M. J.; Li, H.; and Romero, J. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics (TOG)* 36(6): 194.

Lin, J.; Yuan, Y.; Shao, T.; and Zhou, K. 2020. 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*, 5891–5900.

Liu, S.; Li, T.; Chen, W.; and Li, H. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, 7708–7717.

Nazeri, K.; Ng, E.; Joseph, T.; Qureshi, F.; and Ebrahimi, M. 2019. Edgeconnect: Structure guided image inpainting using edge prediction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 0–0.

Paysan, P.; Knothe, R.; Amberg, B.; Romdhani, S.; and Vetter, T. 2009. A 3D face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, 296–301. Ieee.

Ravi, N.; Reizenstein, J.; Novotny, D.; Gordon, T.; Lo, W.-Y.; Johnson, J.; and Gkioxari, G. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* .

Shi, T.; Yuan, Y.; Fan, C.; Zou, Z.; Shi, Z.; and Liu, Y. 2019. Face-to-Parameter Translation for Game Character Auto-Creation. *arXiv preprint arXiv:1909.01064* .

Shi, T.; Zou, Z.; Yuan, Y.; and Fan, C. 2020. Fast and Robust Face-to-Parameter Translation for Game Character Auto-Creation. In *AAAI*.

Tewari, A.; Zollhöfer, M.; Garrido, P.; Bernard, F.; Kim, H.; Pérez, P.; and Theobalt, C. 2018. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2549–2559.

Yang, H.; Zhu, H.; Wang, Y.; Huang, M.; Shen, Q.; Yang, R.; and Cao, X. 2020. FaceScape: a Large-scale High Quality 3D Face Dataset and Detailed Riggable 3D Face Prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, Y.; Deng, J.; Kotsia, I.; and Zafeiriou, S. 2019. Dense 3d face decoding over 2500fps: Joint texture & shape convolutional mesh decoders. In *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition, 1097–1106.