A Video Based Personalized Face Model Generation Approach For Network 3D Games

Xiangyong Zeng, Jian Yao, Mandun Zhang, and Yangsheng Wang

Institute of Automation, Chinese Academy of Sciences,
100080, Beijing, P.R. China
{xyzeng, wys}@mail.pattek.com.cn

Abstract. We have developed a fast generation system for personalized 3D face model and plan to apply it in network 3D games. This system uses one video camera to capture player's frontal face image for 3D modeling and dose not need calibration and plentiful manual tuning. The 3D face model in games is represented by a 3D geometry mesh and a 2D texture image. The personalized geometry mesh is obtained by deforming an original mesh with the relative positions of the player's facial features, which are automatically detected from the frontal image. The relevant texture image is also obtained from the same image. In order to save storage space and network bandwidth, only the feature data and texture data from each player are sent to the game server and then to other clients. Finally, players can see their own faces in multiplayer games.

1 Introduction

The use of personalized 3D face models is a highly desired feature in today's computer games, multimedia titles, medicine, etc. One of the most noticeable trends is the boom toward photorealistic looking and true-life feeling network 3D games, the players want to see their own face on the body of their character. The 3D model of a game character's face is generally composed of a 3D triangular mesh, referred to as the geometry mesh, and an associated composite image of the character's face, referred to as the texture image.

According to MPEG-4 standard [1], which represents the geometry, texture, and animation properties of a 3D face model, the modeling methods for generating a 3-D face model can be generally classified as those that involve: (1) a fully manual approach, (2) a semi-automatic approach and (3) a fully-automatic approach. Fully manual approaches are labor intensive and time consuming, in which every triangular patch of the 3D mesh has to be manually mapped onto the image of the face according to the facial features.

In this paper, we propose an easy-to-use and cost-effective semi-automatic approach for generating the 3D face model for ordinary game player, with a common video camera. In the remainder of this paper, we review and compare the related work in section 2. Section 3 provides an executive summary of our system. Section 4 describes our technique for extracting facial feature points from video image. Section 5 describes deformation algorithm and texture generation.

A practical scheme for applying our system in network 3D game is described in Section 6. Finally, we give the summary and discussion in Section 7.

2 Related work

Fully automatic facial modeling approaches drastically reduce the time required to map a texture onto a 3D mesh. However, while hardware based fully automatic approaches [2][3] are too costly to ordinary user, software based fully automatic approaches [4][5] are very sensitive to the image data and thus may not consistently produce accurate 3D face models. In addition to a 3D face mesh, a composite image that contains facial image data captured from various views also needs to be constructed.

Semi-automatic facial modeling approaches [6][7][8] rely on automatically detecting or manually marking certain features on the face image, such as eyes, nose, and mouth, and initialize the 3D mesh by a global affine warping of a standard 3D mesh based on the location of the detected facial features. However, a global affine transformation generally does not match all local facial features. Thus, the locations of the nodes are fine-tuned in a manual process for each person. Zhang et al. [6] developed a system of reconstructing faces from video sequences with an uncalibrated camera. They extract 3D information from one stereo pair, and then deform a generic face model. Camera poses are computed for the rest of the sequence, and used to generate a cylindrical texture. With model-based bundle adjustment, they can generate highly realistic face models, but they need a long time (about 6 to 8 minutes on a 850MHz Pentium III machine), which is totally unacceptable for ordinary users.

Our work is mainly inspired by the idea that, conveniently and rapidly generating personalized character face in 3D games for the ordinary player. Only in this way the personalized characters can be really popularized in 3D game. Another fact is that player is't very nit-picking to the accuracy of reconstructed face model, if the effect is good enough. So, we use only one single video image captured by one common webcam to generate personalized face model. Comparing with other similar approaches, our facial feature detection and model deformation methods are more simple and quick, which can accomplish the task within one minute on a current standard PC with minimal manual operation.

3 System overview

Firstly, the 2D positions (x- and y-coordinates) of feature points from one frontal face image, captured by the camera, are automatically detected by an improved active shape models (ASM) method [9]. With optional manual operation, these feature points can be fine tuned. Secondly, they are devoted to deform an original game character face mesh with corresponding feature points using an interpolating function based on Radial Basis Functions (RBF) [13]. The z-coordinates of the personalized facial feature points are directly derived from the original mesh, because we only use one single frontal face image. We also propose a simpler technique for generating the texture image by segmenting the elliptical face area from the same image. To generate personalized characters for network 3D games, a practical scheme is also presented. Figure 1 outlines the overview of our system.

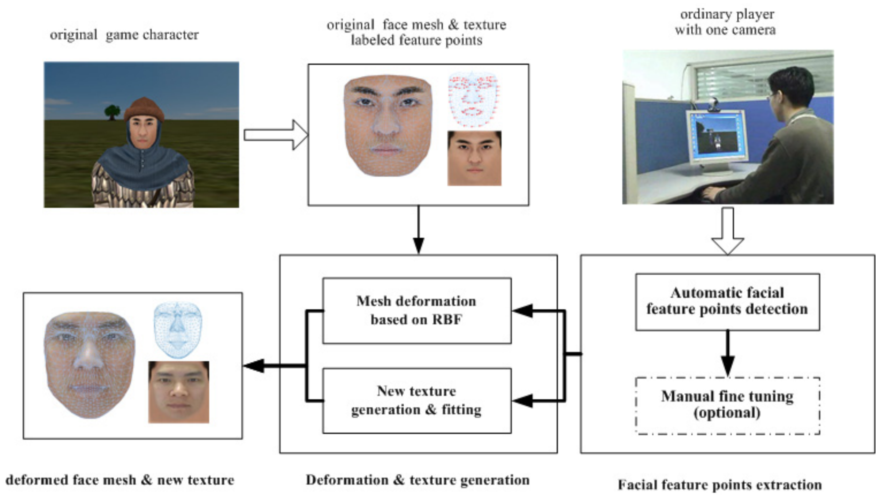

Fig. 1. System overview

The number of triangular nodes and patches in most facial models in network 3D games both could range from several hundreds to several thousands. In this paper, we label 79 feature points on original face model according to MPEG-4 standard, as illustrated in Figure 2. Figure 2 shows an original facial model with a mesh made of 640 nodes and 1430 patches, and a texture(256x256 pixels).

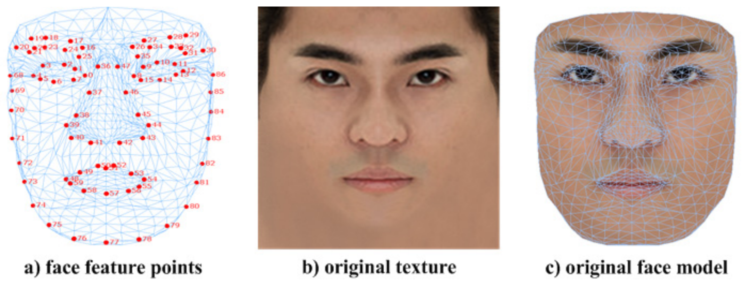

Fig. 2. An original facial model labeled feature points

4 Facial feature points extraction

We used an improved ASM method to automatically detect human face in every video frame and locate the 79 feature points in real-time, as shown in Figure 2. The detected feature points are painted on the same frame image to allow the user consider whether the detected results are suitable or not. By turning head or adjusting camera slightly, the user can get better detected results before pause the auto-detection process. Then, a manual fine tuning step is offered to the user, by dragging painted feature point to new position. This step is optional, and the user can perform it or just skip it.

4.1 Automatic feature points detection

We propose a novel face alignment algorithm, in which local appearance models of facial feature (key) points are constructed using statistical learning. The ambiguity between the truth position and its neighbors requires that the local likelihood model should be able to correctly rank the likelihood of these ambiguous positions.

We adopt RealBoost [10] to learn the ranking prior likelihood models that not only characterize the local features of a ground truth position, but also preserve the likelihood ranking order between the ground truth position and its neighbors. Instead of using principle components analysis (PCA) and one pixel Gabor coefficients, we use Gabor features [11] of key point and its neighbors, which provide rich information to model local structures of a face, to construct "weak" ranking evaluation function set. RealBoost is adopted to solving the following three fundamental problems in one boosting procedure: (1) learning effective features from a large feature set, (2) constructing weak classifiers each of which is based on one of the selected features, and (3) boosting the weak classifiers into a stronger classifier. More details could refer Huang's paper [12].

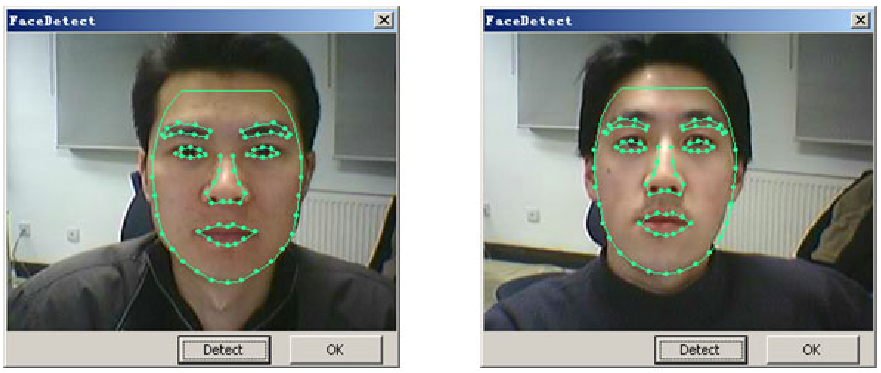

Fig. 3. Results of auto-detecting facial feature points

4.2 Manual fine tuning(optional)

By dragging painted feature points in face image to new positions using mouse, the user can get more accurate location results of feature points. The feature point locations after manual fine-tuning are shown in Figure 4.

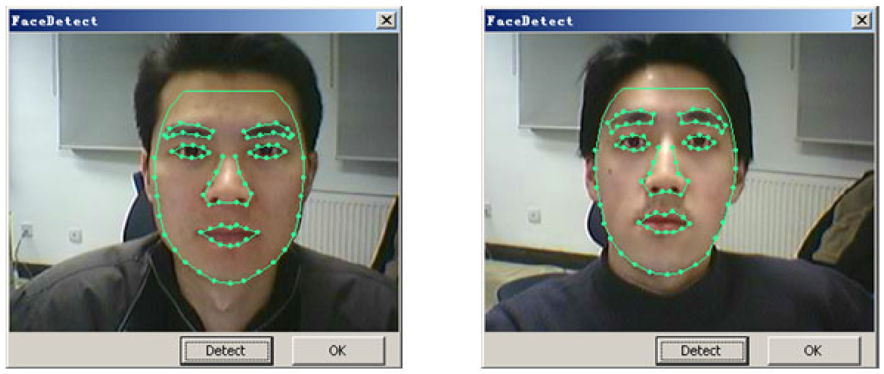

Fig. 4. Results of facial feature points extraction after manual fine tuning

5 Deformation and texture mapping

Given the feature points obtained above, deforming the original face mesh with corresponding feature points is straightforward. At the same time, new texture image is generated from the same face image. After texture mapping, the final personalized face model is generated.

5.1 Deformation

The facial mesh adjustment is a problem about 3D space deformation in fact (In this paper, we only have 2D positions of control points and the z-coordinates for the destination are derived directly from the original mesh). We identify limited control points on it and compute their displacements, then choose an interpolation function which accommodates displacements of control points. Positions of the other nodes are transformed upon the function. At last the final deformation result of the mesh is reached.

There are several choices for how to construct the interpolating function. We use a method based on Radial Basis Functions (RBF) as approximation functions for their power to deal with irregular sets of data in multi-dimensional space in approximating high dimensional smooth surface [13][14]. We choose 2D coordinates of original mesh nodes as embedding space to construct the interpolation function $f(p)$ that suffices displacements of control points:

$$u_i = f(p_i)(0 \leq i \leq N-1) \tag{1}$$

Where u_i are displacements of all control points and N is the number of feature points. We exploit RBF volume morphing to directly drive 2D geometry deformation of face models.

$$f(p_i) = \sum c_i \Phi(\|p_j - p_i\|)(0 \leq j, i \leq N-1) \tag{2}$$

Where $\Phi(\|p_j - p_i\|)$ are RBF and the coefficients c_i are the vector coefficients of control points. We compute equation 2 and get the vector coefficient c_i of every control point. Displacements of other non-feature points may be computed in the form:

$$u = \sum c_i \Phi(\|p - p_i\|)(0 \leq j, i \leq N-1) \tag{3}$$

In this paper, we have chosen to use $\Phi(\|p - p_i\|) = e^{-\|p-p_i\|/64}$.

5.2 Texture generation

The new texture image is generated by the following two steps: 1) segmenting an elliptical face area from the frontal face image according to the feature point positions; 2) scaling and merging it into a background image including side face information from the original texture. The feature points in the frontal face image compose an approximate elliptical contour, which can be used to segment the actual face area from the whole image. In this paper, we assume that the new texture and original texture has the same size. The segmented area must be scaled to a proper size, because the actual face area maybe has different size in different video frame.

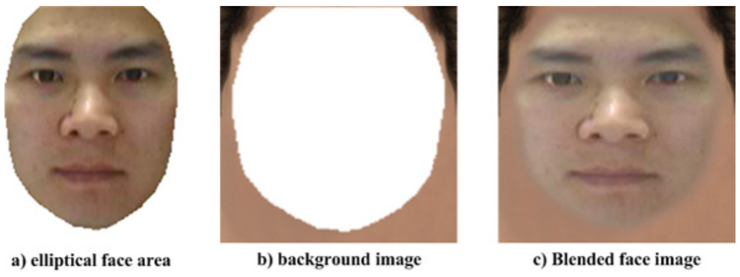

Fig. 5. Texture generation

In addition, we must merge the scaled face image into a background image, which includes synthetical side face texture, because we only have the frontal face information. In practice, the side face information in the background image is copied from the original texture. With smooth template operation, the face contour becomes enough blurring. The result of blended texture is shown in Figure 5.

5.3 Texture fitting

The size of the image is $w \times h$. The maximum x value (x_{max}) is $86.x$ and the minimum x value (x_{min}) is $68.x$, where . The maximum y value $y_{max} =$ YSCALE $\times$ 18.y and the minimum y value (y_{min}) is $77.y$, where YSCALE is a scaling parameter. The 3D coordinates of every point are x, y and z. We mark texture coordinates of every point with x_{tex} and y_{tex}. The left and down corner of frontal face image is (x_t, y_t). According to linear interpolation method, if one point is projected into the front, the texture coordinates of every point are:

$$x_{tex} = x_t/w + (x - x_{min})/(x_{max} - x_{min}) \tag{4}$$

$$y_{tex} = y_t/h + (y - y_{min})/(y_{max} - y_{min}) \tag{5}$$

As for texture coordinates of left and right sides of face, we assign area near the frontal face image. They can be derived from the similar method as above.

5.4 Results

On a current standard PC, for example, a 2.2GHz Pentium IV machine, our system can generate a personalized face model in no more than one minute. Some of this time is spent on manual operation. Several detailed personalized face models in different views are shown in Figure 6 where input frontal face images are shown in Figure 4. They have proper shape and texture.

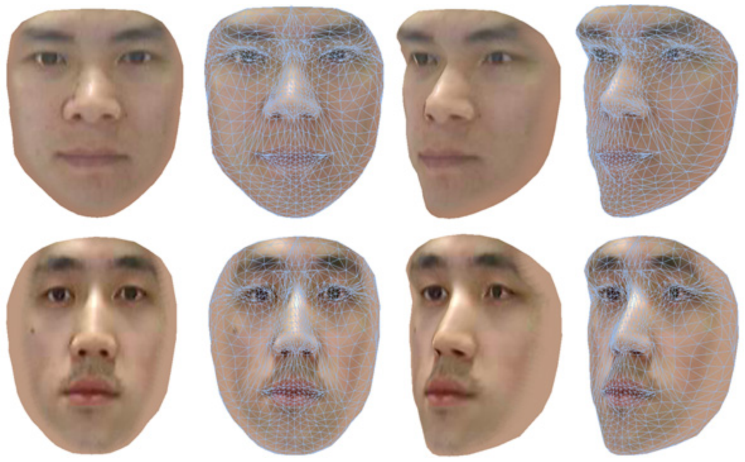

Fig. 6. Results of personalized face models

6 Practical application scheme

Figure 7 outlines a practical application scheme of using our Personalized Face Modeling System (PFMS) in already existing or new designing network 3D games. For applying the scheme, there are three basic demands to the special network 3D game: (1) the original face model files can be parsed and modified expediently, (2) the face feature points of the original face models have been labeled, (3) the game server allows game clients to add Personalized Face Data (PFD) for their character in game, namely feature point positions and new texture image in this paper.

Firstly, the game host or server will confirm whether has already existed it's PFD in game DB server, when a game client logs on the game world. If existed, this client can choose to download it from DB server or update it again by starting PFMS. The PFD, downloaded or newly generated by PFMS is used to construct the Personalized Face Model (PFM) in game clients. In the case of starting PFMS, while mesh deforming and texture mapping, the generated PFD is been upload to DB server for next time use. Instead of generated PFM, only feature and texture data from each player are sent to the game server and then to other clients, in order to save storage space and transmission bandwidth of network 3D games. Finally, players can see their own faces in multiplayer games.

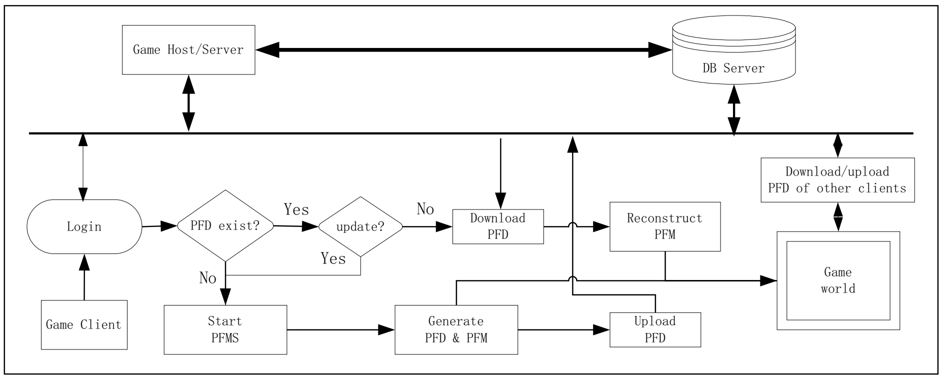

Fig. 7. A practical application scheme of our system

A simple 3D game demo developed by us, named ***ChangeFace***, can give an intuitionistic impression of our PFMS. The video and executable program can be found at our group's website

`http://hci.ia.ac.cn/JointLab/onlinegame-En.htm`
the new version of this demo fully applying this scheme will be coming soon.

7 Summary and discussion

We have developed a system to generate personalized facial model from only one single video image based on facial feature auto-detection. With a few simple clicks for fine tuning by the user, our system quickly generates a person's face model. The experiment results show that it's a simple, easy-to-use and economic approach for end-user at home with a common video camera. These face models can be used, for example, as personalized characters in video games, net meeting, virtual conference, etc. The practical application scheme of the system for network 3D games has given a typical way to achieve these goals.

More accurate facial feature detection and alignment algorithms are need for obtaining reliable auto-extraction results and reducing the sequent manual adjustment, since there are always exist some unfavoured factors, such as lower quality cameras, hostile lighting conditions, as well as people of different races, of different ages, usually have different face shapes and skin colors. We are investigating techniques to improve the alignment algorithm by using more learning samples and solving some lighting problems.

The current character face mesh in most 3D games is relative sparse, our system can work very well in very short time. Considering the future's game applications with higher mesh resolution and details, our system should pay more attention to generate more fine face model, and add some new functions, for example, expression-driven facial animation.

References

1. Abrantes, G. A. and Pereira, F.: MPEG-4 Facial Animation Technology: Survey, Implementation, and Results. IEEE Trans. on Circuits and Systems for Video Technology, vol.9, no.2 (1999) 290–305
2. Sun, W., Hilton, A., Smith, R. and Illingworth, J.: Building Layered Animation Models from Captured Data. Eurographics Workshop on Computer Animation (1999) 145–154
3. Hilton, A. and Inningworth J.: Geometric Fusion for a Hand-held 3-D Sensor. Machine Vision Applications, vol.12, no.1 (2000) 44–51
4. Parke, F. I. And Waters, K.: Computer Facial Animation. A.K. Peters, Wellesley (1996)
5. Akimoto, T., Suenaga, Y. and Wallace, R.S.: Automatic Creation of 3-D Facial Models. IEEE Computer Graphics & App (1993) 16–22
6. Z.Liu, Z. Zhang, C.Jacobs, M.Cohen: Rapid modeling of animated faces from video. Journal of Visualization and Computer Animation. 12(4) (2001) 227–240
7. Lavagetto, Fabio and Pockaj, Roberto: The Facial Animation Engine: Toward A High-Level Interface for the Design of MPEG-4 Compliant Animated Faces. IEEE Transactions on Circuits and Systems for Video Technology, vol.9, no.2 (1999) 277–289
8. Escher, M. and Thalmann, N. M.: Automatic 3-D Cloning and Real-Time Animation of a Human Face. Proceedings of Computer Animation, Geneva, Switzerland (1997)
9. T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham: Active shape models: Their training and application. CVGIP: Image Understanding 61 (1995) 38–59

10. R. E. Schapire and Y. Singer: Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory (1998) 80–91
11. J. Friedman, T. Hastie, and R. Tibshirani: Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity (1998)
12. Huang Xiangsheng, Xu Bin, Wang Yangsheng: Shape Localization by Statistical Learning in the Gabor Feature Space. ICSP04, Beijing, China (2004)
13. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin: Synthesizing Realistic Facial Expressions from Photographs. Siggraph proceedings (1998) 75–84
14. D. Ruprecht, R. Nagel, and H. Mller: Spatial Free-Form Deformation with Scattered Data Interpolation Methods. Computers & Graphics 19(1) (1995) 63–71