Feature Detection and Simplification of 3D Face Data with Facial Expressions

Yong-Guk Kim, Hyeon-Joong Kim, In-Ho Choi, Jin-Seo Kim, and Soo-Mi Choi

We propose an efficient framework to realistically render 3D faces with a reduced set of points. First, a robust active appearance model is presented to detect facial features in the projected faces under different illumination conditions. Then, an adaptive simplification of 3D faces is proposed to reduce the number of points, yet preserve the detected facial features. Finally, the point model is rendered directly, without such additional processing as parameterization of skin texture. This fully automatic framework is very effective in rendering massive facial data on mobile devices.

Keywords: Active appearance model, facial feature detection, feature-adaptive simplification, point-based rendering.

I. Introduction

It is known that humans identify one another by their facial shape and color. However, facial features also provide important visual cues for identification. In perceiving human facial expressions, variations of facial features within the given face play a critical role, which makes face modeling a challenging issue in computer graphics and computer vision. Well-constructed face models can be used for many interesting applications, including makeup simulations and 3D avatars controlled by facial expression recognition.

Among the many ways to acquire 3D facial data, 3D laser scanners are widely used to obtain high-resolution shape and color information, though they can be slow and the data may be noisy. Capturing faces instantly, including such fine features as wrinkles and pores, is possible by combining images taken by multiple cameras [1]. As detailed measured data becomes available to create more realistic digital faces, the amount of data to be processed increases and is likely to include unnecessary and redundant values that will not influence the final images. Therefore, we need a method that simplifies and renders facial data while preserving facial features.

As illustrated in Fig. 1, our three-phase framework is designed to be able to realistically render 3D faces with a reduced set of points. In the first stage, facial features are automatically detected from the projected face in the frame buffer. To detect the eyebrows, eyes, and lips, we develop a new active appearance model (AAM), which is robust against the variation of illumination that can occur in 3D face models reconstructed from multiple images. In the second and third stages, we present a feature-adaptive simplification and rendering scheme that directly works on point-sampled surfaces without any connectivity information between points, which significantly simplifies their processing. In contrast to previous point-based approaches [2], [3], we focus not only on

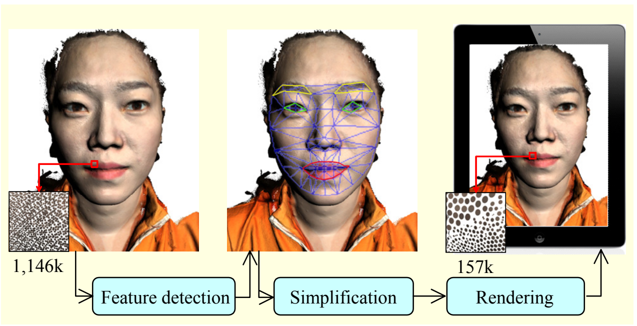

Fig. 1. Proposed framework wherein data flow from feature detection, simplification, and rendering of 3D data is illustrated.

Manuscript received Mar. 10, 2012; revised Apr. 5, 2012; accepted Apr. 24, 2012.

This work was supported by Ministry of Culture, Sports and Tourism and KOCCA in the CT R&D Program 2011 and the National Research Foundation of Korea (2010-0019373).

Yong-Guk Kim (phone: +82 2 3408 3759, ykim@sejong.ac.kr), Hyeon-Joong Kim (hjkim@gce.sejong.ac.kr), In-Ho Choi (anya300@nate.com), and Soo-Mi Choi (corresponding author, smchoi@sejong.ac.kr) are with the Department of Computer Engineering, Sejong Universtiy, Seoul, Rep. of Korea.

Jin-Seo Kim (kjseo@etri.re.kr) is with the Creative Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.12.0212.0112

geometric information but also on color and facial feature information for rendering purposes. Hence, we can considerably reduce the number of points for a given model while preserving detailed geometry and color features.

II. Facial Feature Detection Using a Robust AAM

When an image containing a face is given, the first task is to locate the position of the face, that is, face detection. Locating the face is time-consuming since the entire image must be searched. We adopt AdaBoost, the most popular among many algorithms due to its high speed, even though it requires a long training session. The second task is to detect several facial features concurrently within the detected face. Initially developed to model a face using shape and appearance parameters, the AAM allows us to model the whole face, including facial features. Although it requires an extensive training session using the face database (DB), it operates fast and reliably after completing the modeling process.

Next, the main task of an operating AAM is to synthesize a face model for the given face image. Then, it tries to minimize the error between the given face and the synthesized face through iterations. To synthesize a face, model parameters must be identified using an image alignment algorithm. Such an algorithm normally takes a considerable amount of computing time. The inverse compositional image alignment (ICIA) method [4] is an efficient way to reduce the processing time, as it is possible to track a face in real-time.

The shape in the AAM is represented by meshes generated by connecting landmarks and mathematically defined using vector $\mathbf{s} = \{x_1, y_1, x_2, y_2, \ldots, x_v, y_v\}^{\mathrm{T}}$. The AAM allows linear shape variation. Using such manipulation, we can define it as

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{n} p_i \mathbf{s}_i, \tag{1}$$

in which coefficient p_i is the parameter of the shape, $\mathbf{s}_i$ is the orthonormal vector that is obtained by principal components analysis (PCA), and $\mathbf{s}_0$ is the mean shape.

The appearance (or color) in the AAM is defined by the set of pixels $\mathbf{x}=(x, y)^{\mathrm{T}}$ that lie inside the mean shape $\mathbf{s}_0$. Similar to the shape case, the appearance can be defined by

$$A(\mathrm{x}) = A_0(\mathrm{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathrm{x}), \tag{2}$$

in which the coefficient λ_i is the appearance parameter and A_i is the orthonormal vector that is also obtained by the PCA method. The mean appearance is A_0.

$$\mathrm{M}\big(W(\mathrm{x};p)\big) = A(\mathrm{x}). \tag{3}$$

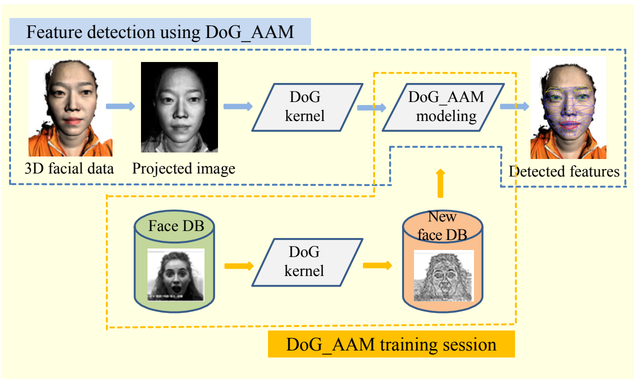

Fig. 2. Building of DoG_AAM and its application for feature detection.

Table 1. Mean errors between ground truth and corresponding fitting data.

	Original AAM	DoG_AAM	Canny_AAM
Mean error	7.57294	4.11276	4.29566

In (3), M represents the model instance. It can be generated by warping the pixel x within the shape **s** by updating the shape **s** using parameter p. Here, the parameter p is calculated using the ICIA method.

Although the AAM is known to be effective in modeling the face or facial expression, the conventional AAM often fails to converge during the fitting process particularly when the illumination condition is not favorable for it. One way to sidestep such a problem is to use either histogram equalization or a discrete cosine transform. However, since our task is to extract the facial features from the face, designing a new AAM that is combined with an efficient edge-detection kernel could be a better choice, as shown in Fig. 2.

To build an AAM using a difference of Gaussian (DoG) kernel, first, a new face DB must be prepared with the face DB and the DoG kernel, as illustrated at the bottom of Fig. 2. Then, an AAM training session using a new face DB is required. Once the DoG_AAM is built, we can use it to detect facial features, as illustrated at the top of Fig. 2, in which 3D facial data is projected on a frame buffer, and the projected 2D image goes through a DoG kernel and the DoG_AAM to obtain AAM fitting. Additionally, we construct a Canny_AAM based on the Canny edge-detecting kernel. We evaluate the performance of the two AAMs using the Yale Face Database, a de facto standard face DB collected under diverse illumination conditions. Results suggest that the DoG_AAM outperforms the Canny_AAM and the original AAM, as shown in Table 1.

A DoG kernel consists of two Gaussian kernels, which have different standard deviations. We can obtain a DoG convolution image by convolving two Gaussian kernels with the given image. The DoG kernel can be defined as

$$D(x,y,\sigma)=\left(G(x,y,\sigma)-G(x,y,\sigma)\right)*I(x,y)$$
$$=L(x,y,k\sigma)-L(x,y,\sigma)\ , \tag{4}$$

in which $L(x, y, k\sigma)$ and $L(x, y, \sigma)$ are Gaussian kernels that have different standard deviations. Considered the most biologically plausible model, it is highly applicable and favored among researchers.

III. Feature-Adaptive Simplification of 3D Faces

The essence of spectral approaches is to measure the effect of a point on the surface by determining its effect on the Laplace-Beltrami spectrum through an approximation of the heat kernel. The measurement and, hence, the algorithms depend on a kernel definition given by $k(x,y)=e^{-\|x-y\|^2}$. A spectral algorithm was introduced in [2] that produces very accurate reconstructions and high quality isotropic or adaptive distributions with blue noise properties at a very low computational and memory cost.

We develop a feature-adaptive simplification scheme for 3D faces by extending the method in [2] to include color adaptive and local sampling features for use in facial rendering. In rendering a face using a simplification method, the facial texture should be preserved to ensure a realistic appearance. Hence, the simplification method needs to allow distributions that adapt to color changes on the surface. In its original form, for spectral samplings, normals are used in addition to positions in the kernel definition such that $\mathbf{x}=[\mathbf{p}/\sigma_p,\ \mathbf{n}/\sigma_n]^{\mathrm{T}}$. We extend the kernel space to include color components so that it becomes $\mathbf{x}=[\mathbf{p}/\sigma_p,\ \mathbf{n}/\sigma_n,\ \mathbf{c}/\sigma_c]^{\mathrm{T}}$, in which σ_c controls the sensitivity to color changes in the model.

We only have color values for the initial points, but values in space are required in resampling. Following the original derivation of the adaptive kernel definition from robust statistics, we use robust local regression on colors to get a value $\mathbf{c}(\mathbf{x})$ for any point $\mathbf{x}$, via the following iterative formula:

$$\mathbf{c}^{k+1}=\frac{\sum_i \mathbf{c}_i k(\mathbf{x}^k,\mathbf{x}_i)}{\sum_i k(\mathbf{x}^k,\mathbf{x}_i)}, \tag{5}$$

in which $\mathbf{x}^k$ includes the current estimate of the color $\mathbf{c}^k$. Starting from the usual kernel approximation $\sigma_c=\infty$, iteration continues until convergence. This gives us a robust color estimate that accurately preserves discontinuities in colors. After computing the new color $\mathbf{c}$, we form the vector $\mathbf{x}$ and proceed with resampling.

As described in section II, the feature detection stage gives the eyes, lips, eyebrows, and facial skin separate labels for weighting. The 2D positions of the weighted pixels are projected back into the 3D positions on the facial surface. The weights w are stored for surface sample points as additional information. For each sample point, the position $\mathbf{p}_i$, normal $\mathbf{n}_i$, color $\mathbf{c}_i$, and weight w_i are stored: $\mathbf{x}_i=(\mathbf{p}_i,\mathbf{n}_i,\mathbf{c}_i,w_i)$.

The final component of our method is utilizing the facial mask to preserve perceptually important features. Denoting the facial mask with $m(\mathbf{x})$ for the point $\mathbf{x}$, the sampling density can be adjusted by weighting the parameter σ_p that controls the spatial extent of the kernel. Using lower values for this parameter will result in finer details and more samples to be placed, so we use the final form $\mathbf{x}=[\mathbf{p}/(\sigma_p w),\ \mathbf{n}/\sigma_n,\ \mathbf{c}/\sigma_c]^{\mathrm{T}}$. The mask weights w depend on the region of the face and are smaller for the important facial features of eyes, lips, and eyebrows. According to our experiment, the simple selection of w=1 for skin and w=0.5 for facial feature areas works well in practice for generating perceptually correct renderings.

For the rendering stage, we use a splat-based representation [5], in which point samples are converted into surface splats with the orientation and spatial extent optimized to fit the geometric and facial features. For efficiency, each splat is represented by a circular disk that is centered near the sample point position $\mathbf{p}$, oriented orthogonal to the normal $\mathbf{n}$, and has the radius $\mathbf{r}$ and the splat color $\mathbf{c}$. The scale factor of the radius $\mathbf{r}$ of each splat is determined in a locally adaptive way. For the sample $\mathbf{x}$, the distance to the closest sample $\mathbf{x}^c$ is computed. Then, the ratio of this distance to the average of all closest distances is used as the local scale factor h_l, given by

$$h_l=\frac{\|\mathbf{x}-\mathbf{x}^c\|}{n^{-1}\sum_{i=1}^{n}\|\mathbf{x}-\mathbf{x}^c\|}. \tag{6}$$

Using this factor along with the global scale factor h_g, we can finally set $\mathbf{r}=h_l h_g \mathbf{r}_0$, in which $\mathbf{r}_0$ is the initial radius.

Although we can guarantee visually continuous appearance of the displayed objects by creating many splats or increasing the size of splats, the number of processed splats and overlapping areas is also increased. The increment of overlapping not only slows down the rendering process significantly but also causes blurring artifacts.

Our surface splatting for final rendering covers more surface area but keeps the overlap as small as possible. To maintain this benefit, especially suited to rendering small-scale feature areas with abrupt color changes, and to attempt to ensure a hole-free representation, we use a hole-filling algorithm in image space instead of excessively increasing splat size in object space. If the size of a hole is sufficiently small (below a threshold value) in the images of albedo color and normal values on the frame buffer object, the color and normal values of the hole are interpolated by the depth-based hole-filling algorithm.

ETRI Journal, Volume 34, Number 5, October 2012 Yong-Guk Kim et al. 793

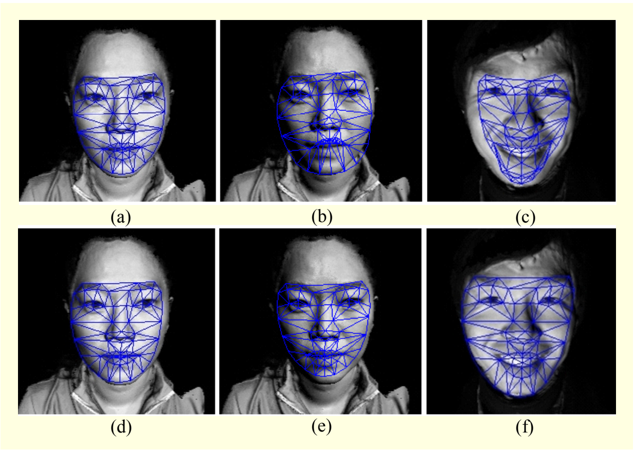

Fig. 3. AAM fittings for face data collected under different illumination conditions: (a)-(c) original AAM and (d)-(f) DoG_AAM.

Table 2. Rendering performance on various devices.

Device	Number of points			
	1,004k	471k	186k	33k
PC (fps)	28	76	120	168
iPad2 (fps)	2	8	14	18

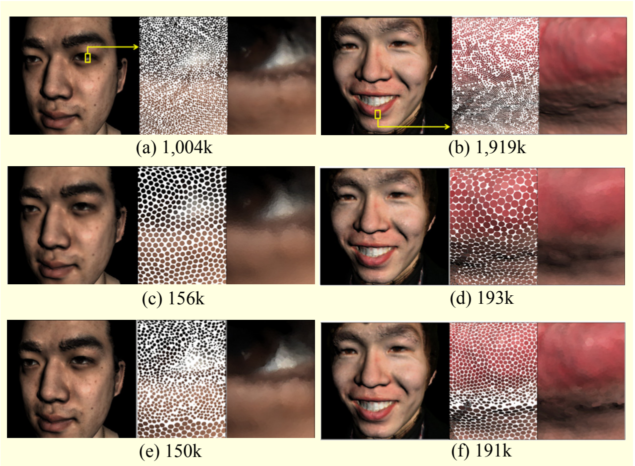

Fig. 4. Comparison of surface splatting methods; (a) and (b) show the results without simplification, (c) and (d) are the results of typical surface splatting using uniform sampling, and (e) and (f) are the results of our feature-adaptive simplification and surface splatting.

IV. Results

Applying the robust AAM combined with the DoG kernel to facial data collected in a well-lit environment results in similar fitting performances by the original AAM and the DoG_AAM, as shown in Figs. 3(a) and 3(d), whereas the DoG_AAM excels when the illumination condition is unfavorable, as shown in Figs. 3(b), 3(c), 3(e), and 3(f). Also shown in Fig. 3(f), our model handles facial expressions very well.

Figure 4 compares rendering results obtained using different surface splatting methods. For reference purposes only, Figs. 4(a) and 4(b) show rendering without simplification. Figures 4(c) and 4(d) show the results of a conventional method [6] that applies a uniform sampling after 85% and 90% simplification, respectively. Figures 4(e) and 4(f) show the results of using our feature-adpative method. In comparison, our simplified cases preserve detailed facial features very well, despite the substantial reduction in total points used.

By varying the number of points, the performances of two different devices are compared. Table 2 shows that rendering the models is feasible on a portable device like iPad2 if the number of points is reduced.

V. Conclusion

We presented a three-phase framework consisting of robust feature detection, feature-adaptive simplification, and rendering 3D facial data. We proposed a DoG_AAM, which detects several facial features concurrently and operates robustly against illumination variations, regardless of facial expression.

Our feature-adaptive approach directly works on point samples without any connectivity information, which allows facial data rendering without surface polygonization and texture mapping, resulting in a much simpler process for high-density point sets. Our framework is particularly effective in retaining perceptually important facial features while substantially reducing the total data size, which is essential for rendering a 3D face on a mobile device.

References

[1] T. Beeler et al., "High-Quality Single-Shot Capture of Facial Geometry," *ACM Trans. Graph.*, vol. 29, no. 4, July 2010, pp. 40:1-40:9.

[2] A.C. Oztireli, M. Alexa, and M. Gross, "Spectral Sampling of Manifolds," *ACM Trans. Graph.*, vol. 29, no. 6, Dec. 2010, pp. 168:1-168:8.

[3] J. Wu and L. Kobbelt, "Optimized Subsampling of Point Sets for Surface Splatting," *Comput. Graph. Forum*, vol. 23, no. 3, Sept. 2001, pp. 643-652.

[4] I. Matthews and S. Baker, "Active Appearance Models Revisited," *Int. J. Comput. Vision*, vol. 60, no. 2, Nov. 2004, pp. 135-164.

[5] H.J. Kim et al., "Subsurface Scattering Using Splat-Based Diffusion in Point-Based Rendering," *Sci. China Inf. Sci.*, vol. 53, no. 5, May 2010, pp. 911-919.

[6] M. Botsch et al., "High-Quality Surface Splatting on Today's GPUs," *Symp. Point-Based Graph.*, June 2005, pp. 17-24.