Face Reconstruction from Shading Using Smooth Projected Polygon Representation NN

Mohamad Ivan Fanany, Masayoshi Ohno, Itsuo Kumazawa

Dept. of Computer Science, Graduate School of Information Science and Engineering
Tokyo Institute of Technology, Japan
West 8E building, Oookayama Meguro-ku, Tokyo 227-0052
email: fanany@kml.cs.titech.ac.jp

Abstract

In this paper, we present a neural-network learning scheme for face reconstruction. This scheme, which we called as Smooth Projected Polygon Representation Neural Network (SPPRNN), is able to successively refine the polygon's vertices parameter of an initial 3D shape based on depth-maps of several calibrated images taken from multiple views. The depth-maps, which are obtained by deploying Tsai-Shah shape-from-shading (SFS) algorithm, can be considered as partial 3D shapes of the face to be reconstructed. The reconstruction is finalized by mapping the texture of face images to the initial 3D shape. There are three interesting issues investigated in this paper concerning the effectiveness of this scheme. First, how effective the SFS provides partial 3D shapes compared to if we simply used 2D images. Secondly, how essential a smooth projected polygonal model is needed in order to approximate the face structure and enhance the convergence rate of this scheme. Thirdly, how an appropriate initial 3D shape should be selected and used in order to improve model resolution and learning stability. By carefully addressing those three issues, it was shown from our experiment that a compact and realistic 3D model of human (mannequin) face could be obtained. This result is ensured through quantitative measurement of average pixel-error and vertex-error between generated model and actual 3D data obtained by 3D scanner device.

1 Introduction

Three-dimensional (3D) face reconstruction is currently receiving a lot of attention in the Computer Vision and Computer Graphics communities. It is a fast growing research field with many application such as virtual reality, animation, face recognition. In all these cases, the recovered model must be compact and accurate, especially around significant areas like the nose, the mouth, the orbits, etc. Since the earliest work in facial modeling to more recent studies [1]-[3], generating realistic faces has been a central goal. However, this remains as a challenging task due to the complex and individual shape of face, and also the subtle and spatially varying reflectance properties of skin.

Artificial neural networks (ANNs), have shown considerable promise in a wide variety of application areas, and have been particularly useful in solving problems for which traditional technique have failed or proved inefficient. Typically, NNs have been shown to be particularly suitable (and have been used extensively) for pattern recognition problems, namely classification, clustering and feature selection. ANNs are also utilized for many other tasks, like optimization, prediction, control, and, more recently, data mining and information retrieval. However, there are still very few studies about the used of NNs for shape representation or surface reconstruction. Rather than directly used NN for reconstruction problem, Braines [4] used NN only to evaluate and verify the reconstruction result of volume rendering technique. Even though NN has been involved further in shape reconstruction process as shown by Cho [5] in the development of a hybrid structure of feed forward NN and radial basis function NN to optimize reflectance model, the recovery of 3D object shape using this optimized model is still be accomplished by ordinary shape-from-shading techniques. There are still not many literature which directly utilized NN for surface reconstruction. Among those literature, Wei [6] proposed a method of shape-from-shading by using radial basis function to parameterize the object depth, and in [7], Iwahori has pursued NN implementations of photometric stereo by estimating the unknown parameters included in the reflectance function and surface gradient distribution using error back propagation learning.

Recently, we developed a unique NN scheme [8] that could store and represent vertices of 3D polygonal object shape. By comparing the projected images of the model (as the outputs of the NN) with the real images taken from different views, the vertices of the object model in 3D space are updated using error back propagation method to approximate the actual 3D object shape. The design setting of updating the NN parameters based on images instead of based on 3D geometrical structures, was inspired by image based rendering techniques [9] that recently caught many attention. These techniques representing a scene as a collection of

images, and new images are generated from real images. Even though these techniques could avoid difficulties in modeling geometry and photometry of the scene, they need the acquisition of a large number of images. Here in this study, provided with only several images, we take the depth-maps from those images by deploying Tsai-Shah SFS algorithm. We treat the depth-maps as partial 3D shapes to be fused by the NN scheme.

There are three distinguished characterizations of the behavior of our scheme, which were not covered in the previous report. First, that is the use of depth maps as partial 3D shapes instead of images. Secondly, there is a significant shortcoming of our previous work; the NN scheme could only generated 'flat' projected polygonal model as its output. We believe that this flat model could only provide a rough and poor approximation of the face to be reconstructed. Hence, in this study we enhanced the NN scheme by deploying a smooth shading module to post-process the output of NN. Thirdly, because a very suitable and potential initial 3D shape, which close to the desired solution could only be generated by expensive and sophisticated techniques, we currently pursue the use of several low-cost initial shape models and compared the result with our previous work.

2 Tsai-Shah SFS

Despite recent advances in depth estimation, e.g., range sensors or stereo vision, SFS remains an important alternative means for estimating shape. This is because, range sensors have limited acting range, and stereo relies heavily on textured regions. SFS complements such methods just in these aspects.

Algorithms for the recovery of shape from shading have been extensively investigated, where the performance of some representative algorithms were compared and analyzed [10]. It was reported that generally, the global minimization techniques yields a very good results, while the local approaches tend to have more error but a lot faster. In this report, we use the Tsai-Shah SFS algorithm [11], which is a very fast technique belong to the local approach, assumes a linearity of reflectance map in term of depth.

Tsai-Shah employed the discrete approximations of p and q, using finite differences in order to linearize the reflectance map in terms of depth Z. The reflectance function for Lambertian surfaces is:

$$R(p_{i,j}, q_{i,j}) = \frac{-s_x p_{i,j} - s_y q_{i,j} + s_z}{\sqrt{1 + p_{i,j}^2 + q_{i,j}^2}}. \quad (1)$$

Using the following discrete approximations for p and q, $p = Z_{i,j} - Z_{i-1,j}$ and $q = Z_{i,j} - Z_{i,j-1}$, the depth map at the n-th iteration, can be solved directly:

$$Z_{i,j}^{n} = Z_{i,j}^{n-1} + \frac{-f(Z_{i,j}^{n-1})}{\frac{d}{dZ_{i,j}} f(Z_{i,j}^{n-1})} \quad (2)$$

The initial estimate of $Z_{i,j}^{0}$ is set to zero for all pixels, Gaussian smoothing is applied to the final depth map to get a smoother result. Note that depth is computed by using a simple division without any matrix inversion (2). This is a simple but efficient algorithm. It was reported that the result of this SFS algorithm works well on smooth lambertian objects with the light source close to the viewing direction. However, it has problems with self-shadows and special care is needed to avoid division by zero.

In this report, the face to be reconstructed is a replica of a smooth human (mannequin) face, assumed to have constant albedo. The object is viewed from three different views with slant angles 0, -45, +45 degrees, with the source location (0.01, 0.01, 1). The images of the face and its corresponding depth-maps (partial shapes) are shown in Fig. 1.

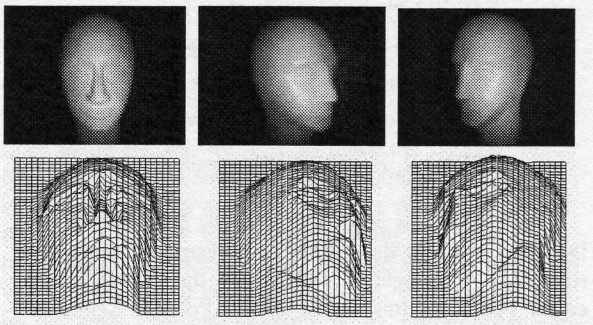

Figure 1. *Three different views and its depth maps*

3 SPPRNN Scheme

Recently, we developed a 3D shape modeling system using the shape's multiple views to determine the polygon parameters [8]. In current development, we deploy Tsai-Shah SFS processing both to the input images and also to the NN output images in order to reconstruct the face based on depth maps instead of images. Since the previous NN scheme could only generate flat shaded model, we set a fast Gouraud smooth shading module to post-process the output of NN. As a whole, we named this enhanced neural network setting as SPPRNN (Smooth Projected Polygon Representation NN). The scheme of SPPRNN for our 3D face reconstruction system is depicted in Fig. 2.

The main part of SPPRNN is PPRNN (Projected Polygon Representation NN), which has three kinds of parameters, i.e., vertex parameters, color parameters, and camera parameters. Before the PPRNN could start to learn, the camera parameters must be set at first. Then the vertex parameters and color parameters are updated

during PPRNN's reconstruction-learning process.

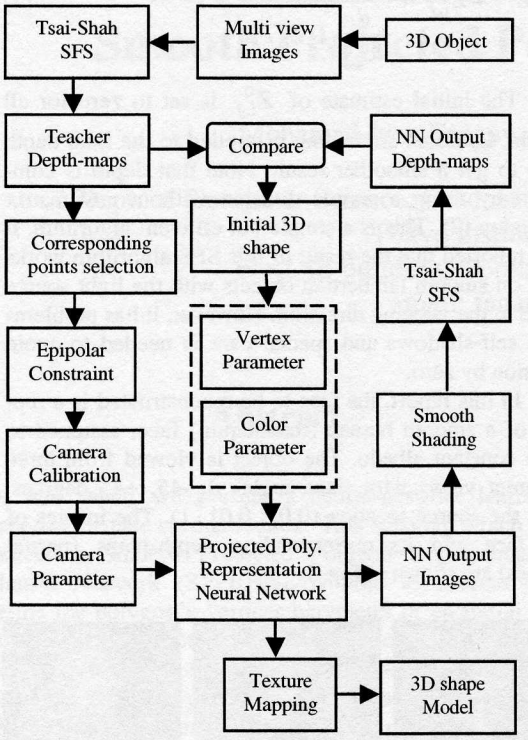

Figure 2. *The SPPRNN learning scheme*

The PPRNN is able to represents 2D shape of polygonal object as its output. A polygon is formed by triangles. A triangle is formed by three straight-line edges. Following this way, a PRNN (Polygon Representation NN) could be constructed by combining a number of TRNNs (Triangle Representation NNs) for triangles representation. Each of TRNN consisted of three ERNNs (Edge Representation NNs) for straight-line edges representation forming the triangle. We could project 3D vertices into 2D plane by attaching the *Project Unit* to the PRNN, hence forming PPRNN.

Basic operation of this PPRNN could be explained as follows. For an input pixel location (x, y) we have to determine the output $F(x, y)$. In determining this output pixel value, we have to find out whether this pixel is inside or outside projected polygon area. If it is outside we defined its output to be $\boldsymbol{g}_{\text{out}}$. If it is inside then we have to find out to which triangle it is belong. After this triangle is found, we have to determine to which side it is belong. Then the output pixel value is assigned to $\boldsymbol{g}_{\text{in}}$. While for every triangle, the output pixel values inside its area were set to $\boldsymbol{g}_{\text{in}}$; the output pixel values outside its area are set to $\boldsymbol{g}_{\text{side}\,j}$ $(j=0,1,2)$ since every triangle adjacently has three neighbors.

In the ERNN, consider $[x, y]^T$ as it's input part, while $\boldsymbol{f}(x, y)$ is output part. $\boldsymbol{v}_0 = [x_0, y_0]^T$, $\boldsymbol{v}_1 = [x_1, y_1]^T$ are called as vertex parameters. A line l is created from the 2 vertices. $\boldsymbol{g}_+ = [r_+, g_+, b_+]^T$, $\boldsymbol{g}_- = [r_-, g_-, b_-]^T$ are called as color parameters. The input of ERNN is image's coordinate value $[x, y]^T$, and the output is a pixel value $\boldsymbol{f}(x, y) = [r, g, b]^T$ at the corresponding coordinate that is defined in equations below:

$$\boldsymbol{l}' = \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} y_1 - y_0 \\ x_1 - x_0 \\ x_1 y_0 - x_0 y_1 \end{pmatrix} \tag{3}$$

$$\boldsymbol{l} = \frac{1}{\sqrt{a'^2 + b'^2}} \boldsymbol{l}' \tag{4}$$

$$\boldsymbol{\alpha} = \boldsymbol{g}_+ - \boldsymbol{g}_- \tag{5}$$

$$\boldsymbol{\beta} = \boldsymbol{g}_- \tag{6}$$

$$d(x, y) = ax + by + c \tag{7}$$

$$s(x, y) = sigmoid(d(x, y), \alpha) \tag{8}$$

$$\boldsymbol{f}(x, y) = s(x, y)\boldsymbol{\alpha} + \boldsymbol{\beta} \tag{9}$$

We attach a *Transform-Unit* to each ERNN. This unit's task is to transform the 2 points $\boldsymbol{v}_0$, $\boldsymbol{v}_1$ to straight-line parameter $\boldsymbol{l} = [a, b, c]^T$. Since the straight-line parameter $\boldsymbol{l}$ is normalized such as $a^2 + b^2 = 1$, $d(x, y)$ of the Eq. (7) state the distance between point (x, y) and straight-line l. The output is near to $\boldsymbol{g}_+$ value at correct-area, and near to $\boldsymbol{g}_-$ value at the false-area.

By combining the structure of three ERNN above, a TRNN could be defined. The parameter setting in TRNN include all ERNN's parameters added with two additional color parameter, i.e., $\boldsymbol{g}_{in}$ and $\boldsymbol{g}_{out}$. For further explanation about TRNN learning, PRNN, and PPRNN, please refer to [8].

3.1 Learning Algorithm

The camera parameters of n-th image $G^{(n)}$, i.e., $\boldsymbol{R}^{(n)}$, $\boldsymbol{T}^{(n)}$, $\boldsymbol{A}^{(n)}$, are computed from correspondence points with other images using epipolar constraint. After the camera parameters: $\boldsymbol{R}^{(n)}$, $\boldsymbol{T}^{(n)}$, $\boldsymbol{A}^{(n)}$, are set, the neural network will throw out the value of its output $\boldsymbol{F}^{(n)}$.

The color parameters $\boldsymbol{g}_j$ are learned as follows. The projection region of triangle $\boldsymbol{T}_j$ is P_j $(\subset R^2)$ $(j = 0,1,2...)$. The region, which is not included in one of triangle projection region, is set as P_{out} $(\subset R^2)$. In each step of learning, $\boldsymbol{g}_j$ is revised to the average pixel value of region P_j in the teacher image.

The vertex parameters $\boldsymbol{V}_k$ are learned using error back propagation rule. After appropriate initial values are given, the error evaluation function E is defined by measuring the difference between depth value in the real (teacher) image $G^{(n)}(x, y)$ with the depth value of neural network output $F^{(n)}(x, y)$. Until this error E is become small enough, $\boldsymbol{V}_k$ are learned. This learning is performed by iteratively compute the gradient descent following equation below:

$$\boldsymbol{V}_k = \boldsymbol{V}_k - \eta \frac{\partial E}{\partial \boldsymbol{V}_k} \qquad (k = 1,2,\cdots,K). \tag{10}$$

Where η learning rate constant, a small positive real value. The evaluation function E is defined as follows:

$$e^{(n)}(x,y) = \left| F^{(n)}(x,y) - G^{(n)}(x,y) \right|^2 \quad (11)$$

$$\varepsilon^{(n)} = \frac{1}{S^{(n)}} \sum_{x,y} e^{(n)}(x,y) \quad (12)$$

$$E = \frac{1}{N} \sum_{n} \varepsilon^{(n)} . \quad (13)$$

Where $S^{(n)}$ is the number of pixel in n-th image. N is the number of images used for learning. $e^{(n)}(x, y)$ is the sum squared error between n-th depth map of neural network $\boldsymbol{F}^{(n)}$ and n-th depth map of teacher image $\boldsymbol{G}^{(n)}$ at location (x, y). The average of this error over all the images is the evaluation function E

From Eq. (10)(11), we could determine

$$\frac{\partial E}{\partial V_k} = \frac{1}{N} \sum_{n} \frac{1}{S^{(n)}} \sum_{x,y} \frac{\partial e^{(n)}(x,y)}{\partial V_k} . \quad (14)$$

If $\partial e^{(n)}(x,y) / \partial V_k$ could be found, the vertex $\boldsymbol{V}_k$ in (8) could be learned. $\partial e^{(n)}(x,y) / \partial V_k$ is back- propagated signal determined using back propagation learning rule, that is the input in backward direction to the output $\boldsymbol{F}^{(n)}$ of neural network. Hence the backward signal is $\partial e^{(n)}(x,y) / \partial \boldsymbol{F}^{(n)}(x,y)$.

For the ERNN learning, the $\partial e / \partial \boldsymbol{v}_k$ could be found by observing the propagation of backward signal after the coordinate value $[\, x,\, y \,]^T$ is set. When $\Delta \boldsymbol{f}$ is put into the output nodes in backward direction, this neural network will operate such as

$$\Delta \boldsymbol{v}_k = \frac{\partial \boldsymbol{f}^T}{\partial \boldsymbol{v}_k} \Delta \boldsymbol{f} \qquad (k = 0,\ 1) \quad (15)$$

is set into the vertices parameters $\boldsymbol{v}_k$ ($k = 0, 1$). Therefore, if the partial input of backward flow is $\Delta \boldsymbol{f} = \partial e / \partial \boldsymbol{f}$, and the vertices parameters $\boldsymbol{v}_k$ ($k = 0, 1$) are updated by backward signal $\Delta \boldsymbol{v}_k$:

$$\Delta v_k = \frac{\partial \boldsymbol{f}^T}{\partial \boldsymbol{v}_k} \Delta \boldsymbol{f} = \frac{\partial \boldsymbol{f}^T}{\partial \boldsymbol{v}_k} \frac{\partial e}{\partial \boldsymbol{f}} = \frac{\partial e}{\partial \boldsymbol{v}_k} \qquad (k = 0,\ 1) \quad (16)$$

At this backward step ERNN operates as Eq. (8), where each unit is passed by backward signal, presented by equations below:

$$\Delta s = \frac{\partial \boldsymbol{f}^T}{\partial s} \Delta \boldsymbol{f} = \alpha^T \Delta \boldsymbol{f} = (\boldsymbol{g}_+ - \boldsymbol{g}_-)^T \Delta \boldsymbol{f} \quad (17)$$

$$\Delta d = \frac{\partial s^T}{\partial d} \Delta s = \alpha\, d(1-d) \Delta s \quad (18)$$

$$\Delta \boldsymbol{l} = \frac{\partial d^T}{\partial \boldsymbol{l}} \Delta d = \tilde{\boldsymbol{x}}^T \Delta d \quad (19)$$

$$\Delta \boldsymbol{l}' = \frac{\partial \boldsymbol{l}^T}{\partial \boldsymbol{l}} \Delta \boldsymbol{l} = \left(\frac{1}{\sqrt{a'^2 + b'^2}} \boldsymbol{I} - \frac{1}{\left(\sqrt{a'^2 + b'^2} \right)^3} \begin{bmatrix} a' \\ b' \\ 0 \end{bmatrix} \right)^T \Delta \boldsymbol{l} \quad (20)$$

$$\Delta \boldsymbol{v}_0 = \frac{\partial \boldsymbol{l}'^T}{\partial \boldsymbol{v}_0} \Delta \boldsymbol{l}' = \begin{bmatrix} 0 & 1 & y_1 \\ -1 & 0 & x_1 \end{bmatrix} \Delta \boldsymbol{l}' \quad (21)$$

$$\Delta \boldsymbol{v}_1 = \frac{\partial \boldsymbol{l}'^T}{\partial \boldsymbol{v}_1} \Delta \boldsymbol{l}' = \begin{bmatrix} 0 & 1 & y_0 \\ -1 & 0 & x_0 \end{bmatrix} \Delta \boldsymbol{l}' . \quad (22)$$

Provided that $\tilde{\boldsymbol{x}} = [x, y, 1]^T$. The backward input signal is:

$$\Delta \boldsymbol{f} = \frac{\partial e}{\partial \boldsymbol{f}} = 2(\boldsymbol{f} - \boldsymbol{G}) \quad (23)$$

For next learning step in TRNN and PRNN, please refer to [8].

3.2 Smooth Model

The color parameters decision at each TRNN, have to follow two constraints. First, the only way to determine the color parameters of the TRNN is to set the parameters based on the *average* intensity of the teacher image across an area, which corresponds to the area of TRNN. Hence, we have to put the output value of PPRNN in the teacher image's rgb frame, where $\boldsymbol{\alpha} = \boldsymbol{g}_+ - \boldsymbol{g}_-$ gives contrast while $\boldsymbol{\beta} = \boldsymbol{g}_-$ gives base-intensity as shown in Eq. (5-6).. Secondly, in order to present hard straight-lines using ERNN, we have to make the sigmoid function to approach a step function (certainly, without loosing its continuous and differentiable property that are needed by PPRNN learning). Following these two inevitable constraints, we could only assign a single intensity value to the whole area of a triangle. A reasonable way to alleviate this shortcoming is to post-process the PPRNN output intact. Therefore, we developed a smooth (Gouraud) shading module, which linearly interpolate the color parameter inside a triangle based on the color at the vertices.

3.3 Initial 3D shape

Currently, we pursue the used of low-cost initial 3D model without user's intervention. We used and compared globe-model and icosahedron-model as initial 3D shapes. Both model are low-cost since their vertices could directly be set analytically. In the purpose of using this model for our initial 3D shape using SPPRNN, the globe model has three drawbacks. First, the faces are unequal in size and have different shapes. Second, it has anisotropic appearance. Third, there are some possibly unresolved areas at the poles that can cause the SPPRNN to be unstable. This may even get worst when we tried to increase the model resolution. Hence, further model resolution enhancement becomes impossible. The projected polygonal model using higher resolution could be generated if we use icosahedron model.

4 Quantitative Evaluation

The evaluation of the effectiveness of the SFS technique in providing partial 3D shape of the face to be reconstructed, need to be quantitatively measured. For this purpose, we extract the actual 3D shape as reference, using 3D scanner device. We measure average vertex-error and pixel-error of 3D reconstructed model by using SFS (depth-based) and also without using SFS (images-based), compared to the actual 3D data. Both vertex-error and pixel-error simultaneously define the term of 'realistic ness' of the reconstruction. Since this data has different scale, resolution, and alignment, an appropriate normalization should be performed before evaluation process

The measurement of average vertex error is performed as follows. Since vertex resolution of 3D reconstructed result is much less than actual 3D data, there is no one-to-one correspondence between those vertices. This make the quantitative measurement become uncertain and difficult. In our present study, for every vertex in the reconstructed model, first we have to find the nearest vertex in the actual 3D data. We assumed that the reconstructed face should be convex, and there is only one closest reference vertex for every vertex of the model. Then we could measure the average vertex-error as:

$$E_v = \frac{1}{N_m} \sum_{i=1}^{N_m} \sum_{j=1}^{N_a} (v_i - V_j)^2 \qquad N_m < N_a \quad (24)$$

$$\| v_i - V_w \| = \min_j \| v_i - V_j \| \quad (25)$$

where N_m and N_a, are number of vertex in the reconstructed model and actual 3D data respectively. V_w are the nearest model's vertex to the actual 3D data v_i.

5 Result of Experiment

The first experiment was to measure the effectiveness of SFS compared to the reconstruction result without using SFS (plain images). Images from reconstructed face after 1000 iterations viewed from two different poses are presented in Fig. 3. Using Tsai-Shah SFS was experimentally shown a more compact and realistic reconstructed face. Even though there are still some little defects around nose, the areas around chin, forehead, and orbits are more accurately joined. Presently, we have also analyzed and compared several SFS algorithms, e.g., Bischel-Pentland (propagation) and Zheng-Chellappa (minimization) approaches. This analysis suggests some encouraging results.

The qualitative measurement above is assured by following quantitative evaluation of three aspects (Table 1), average pixel-error, average vertex-error, and also their computational time needed for total reconstruction. The unit scale for pixel-error is unit pixel scale between 0 – 255.0. The unit scale for vertex-error is unit scale between 0 – 1, where the vertex-error 1 indicating the farthest vertex position from origin. The computational time is measured in minutes scale.

	Image-based	SFS-based
Average pixel-error	38.7	28.0
Average vertex-error	0.074	0.014
Computational Time	25	26.5

Table 1. *Quantitative measurement on 3D reconstruction results after 1000 iterations*

The second experiment was to measure the effectiveness of smooth shading post-processing. Since now we have two kinds of output, i.e., flat output and smooth output, we proposed three schemes to measure this smooth shading. First is to use only flat output in the learning of the PPRNN. We called this first scheme as 'without smooth' scheme. Secondly, we use both flat and smooth output, then we called this scheme as 'with smooth' scheme. Thirdly, we use only smooth output and we called this as 'only smooth' scheme. The error profile of those three schemes, are presented simultaneously in Fig. 4. Confirmation through reconstruction result after different iteration is given in Fig. 5.

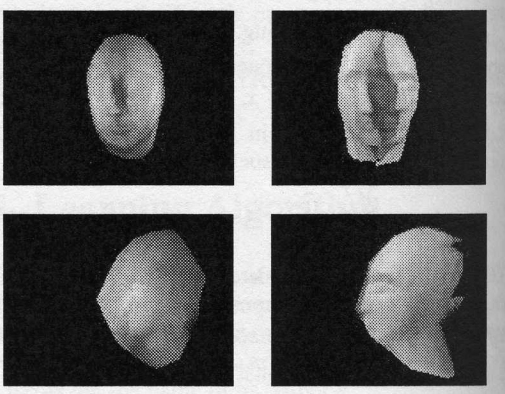

Figure 3. *Result of reconstruction with SFS (left images) and without SFS (right images)*

It was shown from the error profile Fig. 4, that 'with smooth' scheme could converge faster compared to 'without smooth' scheme. However, the 'only smooth' profile caused instability after around 200 iterations and it also shown hard to converge. These results were exactly confirmed by the reconstruction results in Fig. 5.

The third experiment was to measure the effectiveness of icosahedron-model in stabilizing the SPPRNN learning. It was shown from Fig. 6, that icosahedron-model drive the system to converge faster, and more stable, even though involved less number of triangles and vertices.

6 Conclusion

A new approach for reconstruction of face from shading using a unique NN design has been described. Using this approach, a fast and realistic face modeling could be achieved, even though only a very limited set of images is available. We conducted three measurements on the effectiveness of SFS, smooth shading, and icosahedron-model on the this NN learning. In order to obtain a more compact and realistic result, and also faster to converge, and more stable reconstruction system, we need to carefully consider those measurement results. It was both qualitatively and quantitatively shown that SFS could be effectively used to ensure our neural network to converge, which otherwise impossible if we used only 2D images. For future study, we believe that the smoothness constraint as the basic assumption of SFS techniques, should be effectively conformed by the underlying representation neural network. For this reason, currently we are extending this system for smooth surface patch representation, e.g., Bezier and Spline.

References

[1] T. Akimoto, Y. Suenaga, "Automatic Creation of 3D Facial Models", *IEEE Computer Graphics & Application*, pp. 16-22, September 1993.

[2] J.J. Atick, P.A. Griffin, A. N. Redlich, "Statistical Approach to Shape From Shading: Reconstruction of Three-Dimensional Face Surfaces from Single Two-Dimensional Images", Neural Computation Vol. 8, No. 6, August 15, 1996.

[3] R. Lengagne, P. Fua, O. Monga, "3D stereo reconstruction of human face driven by differential constraints", *Image and Vision Computing* 18, pp.337-343, 2000.

[4] S.A. Braines, R.J. Cant, "A framework for the evaluation of vol. rendering tech on a task specific basis using NN", *Computer & Graphics* 25, pp.643-663, 2001.

[5] S. Cho, T. Chow, "Enhanced 3D shape recovery using the Neural-based Hybrid Reflectance Model", *Neural Computation* 13, pp.2817-2637, 2001.

[6] G.Q. Wei, G. Hirzinger, "Parametric Shape-from-Shading by Radial Basis Functions", *IEEE Trans. PAMI,* vol. 19, No. 4, April 1997.

[7] Y. Iwahori, A. Bagheri, R.J. Woodham, "Neural network implementation of photometric stereo", *Vision Interface95*, pp.81-88, Quebec May 1995.

[8] I. Kumazawa, M. Ohno, "3D Shape Reconstruction form Multiple Silhouettes: Generalization from Few Views by Neural Network Learning," A. Arcelli et. Al (Eds.): IWVF4, LNCS 2059, pp. 687-695, 2001

[9] Z. Zhang, "Image-Based Geometrically-Correct Photorealistic Scene/Object Modeling (IBPhM): A Review", *Proc. ACCV'98*, Hong-Kong, 1988.

[10] R. Zhang, P. Tsai, J.E. Cryer, M. Shah, "Shape from Shading: A Survey", IEEE Trans. On PAMI, vol. 21, No. 8, August 1999

[11] P. Tsai, M. Shah, "Shape from Shading Using Linear Approximation", Image and Vision Computing Journal, vol. 12, no. 8, pp.487-488, 1994

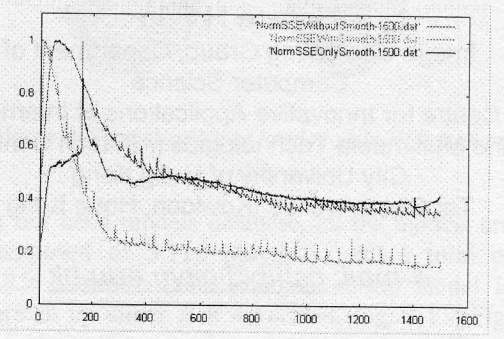

Figure 4. *The normalized SSE profile to measure the effectiveness of smooth shading post-processing*

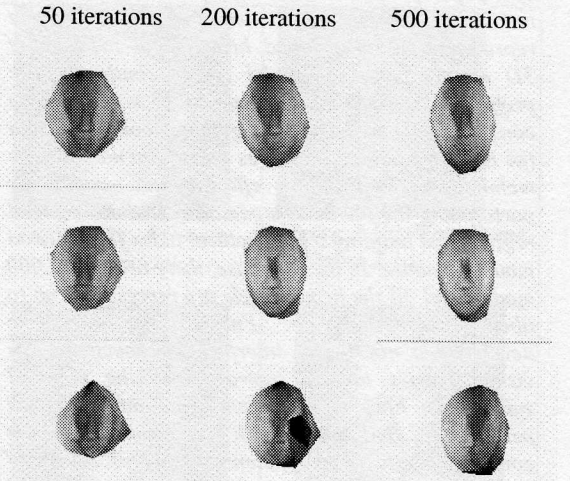

Figure 5. *Reconstruction result of the three schemes, 'without smooth' (upper row), 'with smooth' (middle row), and 'only smooth' (lower row)*

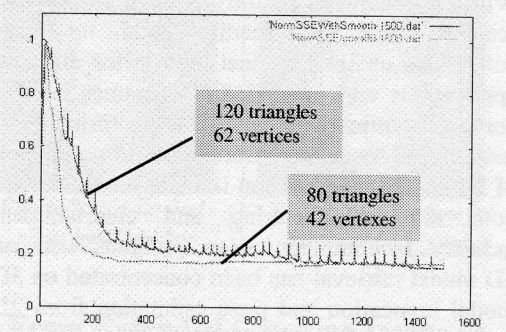

Figure 6. *The error profiles (normalized SSE) comparison between icosahedron and globe models*