Multilevel Quadratic Variation Minimization for 3D Face Modeling and Virtual View Synthesis

Xiaozheng Zhang[1], Yongsheng Gao[1], and Maylor K.H. Leung[2]
[1]*School of Microelectronic Engineering, Faculty of Engineering and Information Technology, Griffith University, Nathan Campus, Australia*
x.zhang@griffith.edu.au, yongsheng.gao@griffith.edu.au
[2]*School of Computer Engineering, Nanyang Technological University, Singapore*
asmkleung@ntu.edu.sg

Abstract

One of the key remaining problems in face recognition is that of handling the variability in appearance due to changes in pose. One strategy is to synthesize virtual face views from real views. In this paper, a novel 3D face shape-modeling algorithm, Multilevel Quadratic Variation Minimization (MQVM), is proposed. Our method makes sole use of two orthogonal real views of a face, i.e., the frontal and profile views. By applying quadratic variation minimization iteratively in a coarse-to-fine hierarchy of control lattices, the MQVM algorithm can generate C^2 *-smooth 3D face surfaces. Then realistic virtual face views can be synthesized by rotating the 3D models. The algorithm works properly on sparse constraint points and large images. It is much more efficient than single-level quadratic variation minimization. The modeling results suggest the validity of the MQVM algorithm for 3D face modeling and 2D face view synthesis under different poses.*

1. Introduction

Existing work in face recognition has demonstrated good recognition performance on frontal, expressionless views of faces with controlled lighting [6,7]. However, it is still a challenge to develop a pose-lighting-expression invariant face recognition system. In this research, we concentrate on the pose-invariant recognition of a novel face, which is viewed from an arbitrary viewpoint with controlled lighting condition and neutral facial expression.

Prior work shows that it is realistic yet challenging to perform pose-invariant face recognition using virtual views. The main difficulty is the task of generating the virtual views of a face from a limited number of real views. A number of models have been proposed for the purpose of virtual view generation. A.S. Georghiades *et al.* [1] generated a 3D illumination cone model from seven frontal face images under slightly different lighting conditions. Using the information of surface and reflectance of the model, the novel face views from different poses and lighting conditions can be synthesized. The limitation of this model lies in the restriction to the lighting sources in the training set. 3D deformable models have been applied to pose-invariant face recognition, too. A wire frame generic model [3,5] is developed, using prior knowledge of human head geometries. The 3D model contains one or more coefficients, which can be adjusted for different individual faces. Both the shape and the texture information of a specific face from a novel viewpoint can be synthesized based on the generic model. However, 3D deformable models suffer from the prototype difference, which happens when the gallery face shapes are far different from the generic models. Under such circumstances, individualized face models can hardly converge to the realistic face shapes.

Another approach to generate virtual views of given poses is to warp real views of the known pose. D. Beymer and T. Poggio [2] introduced prototypical information into the generation of virtual views from a single real view. They mapped 2D facial transformations observed on a prototype face onto a novel, non-prototype face to generate virtual views. The offset view of a face was selected as the real view, since it contains more information than the frontal view. However, in real applications, frontal views rather than offset views are usually available. And the rotation angles are usually limited up to 30 degrees in depth. Neural network is another choice in pose-invariant face recognition. F. Wallhoff *et al.* [4]

Proceedings of the 11th International Multimedia Modelling Conference (MMM'05)
1550-5502/05 $20.00 © 2005 **IEEE**

developed a neural net to synthesize profile views using frontal face information. The weakness of neural networks is that the network architecture has to be extensively tuned, and a large number of face samples are needed for the training of a neural network.

In this paper, we present a novel 3D face shape-modeling algorithm, the Multilevel Quadratic Variation Minimization (MQVM), in order to build individualized face shapes and face views from only frontal and profile views for the pose-invariant face recognition. Simply based on facial features specified on the face views, MQVM algorithm can generate C^2 continuous 3D face surfaces efficiently. Since MQVM algorithm doesn't require any generic model, it is entirely individualized and free from prototype difference. It requires only two face views of each person and it is promising in the face recognition applications where few face views are available.

The rest of the paper is organized as follows. Section 2 introduces quadratic variation minimization for 3D surface approximations. An improved multi-level quadratic variation minimization is described in Section 3. In Section 4, a framework of 3D face modeling is described and the multilevel quadratic variation minimization is applied. The 3D modeling results and a conclusion are given in Section 5 and 6, respectively.

2. Quadratic Variation Minimization

The quadratic variation

$$\Theta(f) = \left\{ \iint \left(f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right) dxdy \right\}^{\frac{1}{2}} \tag{1}$$

is introduced to approximate the 3D surfaces from a set of feature points. It generates C^2 continuous surfaces [12]. Clearly any function, which minimizes the functional $\Theta(f)$ also minimizes the functional $\Theta^2(f)$, and vice versa, provided that the functional is always positive in value. And we use s to denote the surface which we are fitting the feature points. Hence, without loss of generality, one can consider the minimization of

$$\Theta(s) = \iint \left(s_{xx}^2 + 2 s_{xy}^2 + s_{yy}^2 \right) dxdy. \tag{2}$$

In order to determine the structure of the algorithm, one must address the issue of the form of the output representation, since that will have a major effect on the actual algorithm. The continuous functional must now be converted to a form applicable to a discrete grid. Without loss of generality, assume that the grid is of size $m \times n$. At each point (i, j) on the grid, a surface depth may be represented by $s_{(i,j)}$. Each such surface depth may be considered as an independent variable, subject to the constraint of the surface. To discretize functional (2), the following approximations are chosen.

The second partial derivative in the x direction is approximated by

$$s_{xx} = \frac{1}{h^2} \left[s_{(i+1,j)} - 2 s_{(i,j)} + s_{(i-1,j)} \right] + O(h^2) \tag{3}$$

where h is the grid spacing, and $O(h^2)$ indicates that the approximation is valid to terms of order h^2. Similarly, the second partial derivative in the y direction may be approximated by

$$s_{yy} = \frac{1}{h^2} \left[s_{(i,j+1)} - 2 s_{(i,j)} + s_{(i,j-1)} \right] + O(h^2). \tag{4}$$

The cross second partial derivative can be approximated by

$$s_{xy} = \frac{1}{h^2} \left[s_{(i+1,j+1)} - s_{(i+1,j)} - s_{(i,j+1)} + s_{(i,j)} \right] + O(h^2). \tag{5}$$

Having converted the surface function and the differential operators, one must convert the double integral to a discrete equivalent. This can easily be done, by using a double summation over the finite difference operators applied to the discrete grid. Hence, the discrete form of the quadratic variation functional is

$$\begin{aligned} \Theta(s) = & \sum_{i=1}^{m-2} \sum_{j=0}^{n-1} \left(s_{(i+1,j)} - 2 s_{(i,j)} + s_{(i-1,j)} \right)^2 \\ & + \sum_{i=0}^{m-1} \sum_{j=1}^{n-2} \left(s_{(i,j+1)} - 2 s_{(i,j)} + s_{(i,j-1)} \right)^2 \\ & + 2 \sum_{i=0}^{m-2} \sum_{j=0}^{n-2} \left(s_{(i,j)} - s_{(i+1,j)} - s_{(i,j+1)} + s_{(i+1,j+1)} \right)^2. \end{aligned} \tag{6}$$

Finally, the characterization of the constraints must be considered. The case of interpolation will be considered first, where the interpolated surface is required to pass through the known points. Let $\delta = \{(i, j) \mid \text{there is a known depth value at the grid point } (i, j)\}$ be the set of grid points for which a depth value is known. Then the constraints on the optimization problem have the form $s_{(i,j)} - c_{(i,j)} = 0$ for all points (i, j) in the set δ, and where the

$c_{(i,j)}$'s are a set of constants reflecting the stereo data. Therefore, the surface approximation problem is

minimize

$$\sum_{i=1}^{m-2}\sum_{j=0}^{n-1}\left(s_{(i+1,j)} - 2s_{(i,j)} + s_{(i-1,j)}\right)^2$$

$$+\sum_{i=0}^{m-1}\sum_{j=1}^{n-2}\left(s_{(i,j+1)} - 2s_{(i,j)} + s_{(i,j-1)}\right)^2$$

$$+2\sum_{i=0}^{m-2}\sum_{j=0}^{n-2}\left(s_{(i,j)} - s_{(i+1,j)} - s_{(i,j+1)} + s_{(i+1,j+1)}\right)^2,$$

subject to $s_{(i,j)} - c_{(i,j)} = 0$, $\forall (i,j) \in \delta$. (7)

To solve the problem (7), gradient projection method can be considered [10]. One step of the gradient project algorithm is as follows: Given a feasible point $\vec{s}_k$

1. Find the subspace of active constraints M, and form A_q.
2. Calculate $P = I - A_q^T (A_q A_q^T)^{-1} A_q$ and $\vec{d} = -P\nabla\Theta(s_k)^T$.
3. Find α_1 and α_2 achieving, respectively,

   $\max\{\alpha : \vec{s}_k + \alpha\vec{d}$ is feasible$\}$,

   $\min\{\Theta(\vec{s}_k + \alpha\vec{d}) : 0 \le \alpha \le \alpha_1\}$.
4. Update $\vec{s}$ by $\vec{s}_{k+1} = \vec{s}_k + \alpha_2\vec{d}$ and return to 1.

Since all the constraints are equality constraints, they are all active at every iteration. Thus, the matrix A_q is a $q \times mn$ matrix, where q is the number of constraints, and A_q is made up of rows corresponding to the active constraints. It has rows consisting of a 1 in the position corresponding to the grid point (i,j) for $(i,j) \in \delta$ and 0's elsewhere. Therefore,

$$A_q A_q^T = I, \tag{8}$$

and

$$A_q^T A_q = B_\delta, \tag{9}$$

where B_δ is an $mn \times mn$ matrix consisting of 0's except for those rows corresponding to a point in δ, such rows containing a 1 for the diagonal element. Thus, the projection matrix $P = I - B_\delta$ consists of all 1's except for diagonal elements in those rows corresponding to a constraint point in δ, such elements being 0's. The effect of multiplying the projection matrix P to a vector $\vec{s}$ ($\vec{s}$ is a differential image) is to ignore any components corresponding to given constraint points, while preserving all other components unaltered. By expanding the double summation and performing the differentiation, the direction vector $\vec{d}$ can be determined.

After determining the form of the direction vector, which specifies the direction in which to move in order to reduce the objective function $\Theta(\vec{s})$ and refine the surface approximation, it is necessary to determine the amount to move in this direction, that is, to determine the value of α_2 such that $\Theta\left(\vec{s}_k + \alpha_2\vec{d}\right)$ is minimized. Since the projection matrix P removes all the components in the positions of constraint points in the direction vector $\vec{d}$, $\vec{s}_{k+1} = \vec{s}_k + \alpha_2\vec{d}$ is always feasible while $\alpha_2 \ge 0$. Therefore, the value for α_2 is given by

$$\frac{\partial}{\partial\alpha_2}\Theta\left(\vec{s}_k + \alpha_2\vec{d}\right) = 0. \tag{10}$$

Thus, the surface $\vec{s}$ is refined and the iteration continues until the magnitudes of all components of the direction vector are smaller than some constant ε.

Let $s_{0(i,j)} = c_{(i,j)}$ if $(i,j) \in \delta$, and $s_{0(i,j)} = 0$ otherwise. Fig. 1 shows a cylinder surface approximation produced by a single level quadratic variation minimization. Fig. 1(a) shows the constraint points located on the initial surface. The resolution of the cylinder surface is 256×256. Fig. 1(b) to Fig. 1(e) are the surfaces generated produced by QVM approximation at 10^4 (b), 10^5 (c), 10^6 (d), and 10^7 (e) iterations.

3. Multilevel Quadratic Variation Minimization

As shown in Fig. 1, QVM algorithm converges to a C^2 smooth 3D surface. However, it takes usually more than 1,000,000 iterations to produce an accurate surface. The reason why QVM is slow is that the constraint points are sparse and face images are usually of high resolutions. And the objective function $\Theta(\vec{s})$ is a neighboring functional taking effect only on local areas of points.

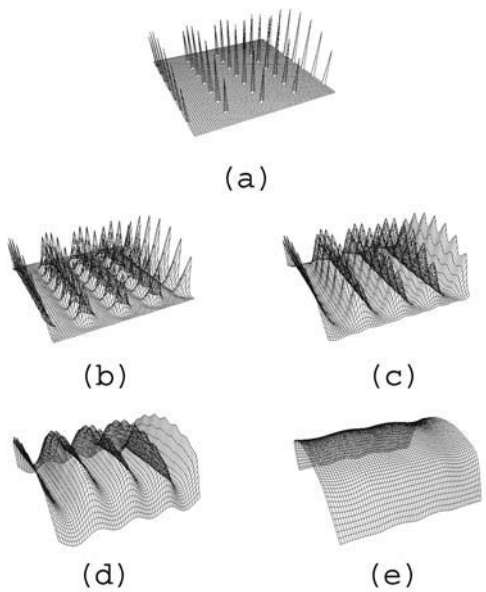

Fig 1. Surface approximation produced by QVM.

In this section, we present a multilevel quadratic variation minimization (MQVM) technique that overcomes the drawback of QVM. In MQVM, a coarse-to-fine hierarchy of control lattices, Φ_0, Φ_1, …, Φ_w, is used to derive a sequence of surface refinements with QVM approximations. Let h_k be the spacing between control points on the lattice Φ_k. We assume that h_0 and $h_w = 1$ pixel are given and $h_{k+1} = \frac{1}{2} h_k$. When the surface $\vec{s}$ is approximated with a coarse control lattice, the constraints points merge with each other and result in a smooth approximation, although they are not exactly satisfied individually. And the result surface is used to provide the initial surface $\vec{s}_0$ in the next level.

In MQVM, a sequence of QVM approximations starts with the coarsest control lattice Φ_0, and ends with the finest control lattice Φ_w, which is the actual resolution of the surface $\vec{s}$. With a control lattice Φ_k, all the constraint points in the same grid merge into one constraint point and its depth is the average. The QVM approximation iterates until the magnitudes of all components of the direction vector are smaller than some constant ε. Then the next finer control lattice Φ_{k+1} is used for successive QVM approximations, as long as Φ_k is not the finest control lattice. The following pseudocode outlines the MQVM algorithm. In this algorithm, QVM approximations are successively applied with a hierarchy of control lattices to make the 3D points in $\vec{s}$ gradually approach their positions in 3D space from their initial positions. The approximation procedure of MQVM algorithm is the composition of several approximations derived by QVM. Therefore, MQVM generates a C^2 continuous surface.

Algorithm MQVM

Input: constraint point set $C = \{c_{(i,j)} | (i,j) \in \delta\}$

Output: 3D surface $\vec{s}$

let Φ be the coarsest control lattice (e.g., 2×2)

let h be the spacing in the control lattice Φ

let δ' be the constraint points in Φ

let $C' = \{c_{(i,j)} | (i,j) \in \delta'\}$ be the constraint point set under Φ

initialize the surface $s_{(i,j)} = \begin{cases} 0, (i,j) \notin \delta' \\ c_{(i,j)}', (i,j) \in \delta' \end{cases}$

while Φ is not the finest control lattice **do**

let Φ be the next finer control lattice

let h be the spacing in Φ

update δ' with new control lattice Φ

update C' under Φ and δ'

refine $\vec{s}$ from C' by QVM approximation under Φ

end

Fig. 2 gives an example in which the MQVM algorithm is applied to generate a cylinder surface given the constraint point set. Fig. 2(a) is the initial stage of the surface $\vec{s}$ with only constraint points located. Fig. 2(b) through Fig. 2(f) shows a sequence of surface approximations from successive QVM algorithms. The resolution of the surface is 256×256, and the control lattices of the surface approximations are 4×4 (b), 8×8 (c), 16×16 (d), 32×32 (e), and 256×256 (f) respectively. It takes 10,000 iterations for each resolution level and totally 70,000 iterations to produce the 3D face surface. Compared with the single-level QVM approximation described in section 2, MQVM is much more efficient, especially when the surface resolution is high and the constraint points are sparse.

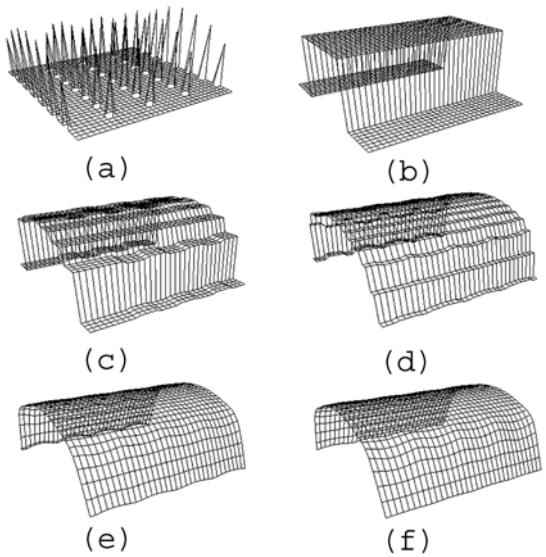

Fig 2. A sequence of approximations produced by MQVM algorithm.

4. 3D face surface generation

An entire 3D face surface generation consists of face view normalization, facial feature specifications, and the multilevel quadratic variation minimization approximation.

Given a frontal view and profile view of a face, to bring the facial organs (i.e., eyes, nose and mouth) to lie on the same levels, both the frontal view and profile view need to be normalized. For the frontal face view, the locations of left and right eyes are used as control points. It is because eyes are the most stable features of human face among all the facial features. Before frontal view normalization, the locations of eyes are determined. Since there are a number of automatic eye detection algorithms [11], we don't intend to tackle this task again and the eyes are manually located. We assume that the frontal face view is free from rotation in depth. In the process of frontal view normalization, the two eyes are tuned to the same height by a 2D rotation in the image plane. Then the frontal view is scaled and translated to let the two eyes locate at two predefined positions.

Let $\vec{e}_{1,l} = (x_{1,l}, y_{1,l}, 1)^T$ and $\vec{e}_{1,r} = (x_{1,r}, y_{1,r}, 1)^T$ denote the left and right eyes' positions in the original frontal face view in homogenous space. The predefined positions are $\vec{e}_{1,l}' = (x_{1,l}', y_{1,l}', 1)^T$ and $\vec{e}_{1,r}' = (x_{1,r}', y_{1,r}', 1)^T$. The 2D transformation matrix M_1 is

$$M_1 = \begin{pmatrix} k_1 \cos\alpha_1 & k_1 \sin\alpha_1 & x_1 \\ k_1 \sin\alpha_1 & -k_1 \cos\alpha_1 & y_1 \\ 0 & 0 & 1 \end{pmatrix}, \quad (11)$$

where k_1 is the scaling factor, (x_1, y_1) is the displacement of the translation, and α_1 is the rotation angle. Based on the original and predefined locations of the two eyes, a unique frontal normalization transformation is determined by

$$(\vec{e}_{1,l}', \vec{e}_{1,r}') = M_1(\vec{e}_{1,l}, \vec{e}_{1,r}). \quad (12)$$

Every point on the frontal view $(x_1, y_1)'$ can be transformed to the new position $(x_1', y_1')'$ in the normalized frontal view under a linear 2D transformation by

$$(x_1', y_1', 1)^T = M(x_1, y_1, 1)^T. \quad (13)$$

Similar to the frontal view normalization, we locate the positions of the eye (only one of the two eyes is visible in the profile view), the nose top and the center of the mouth. The transformation matrix of the profile normalization M_2 is of the same form with M_1. The only difference is that profile view is normalized according to the normalized frontal view provided the positional information of the facial features. The *y*-coordinates of the facial features are the same in the normalized frontal view and profile view. For example, let $\vec{e}_{2,l} = (x_{2,l}, y_{2,l}, 1)^T$ denote the original eye position in the profile view and $\vec{e}_{2,l}' = (x_{2,l}', y_{2,l}', 1)^T$ denote the position in the normalized profile view. The constraint $y_{1,l}' = y_{2,l}'$ is applied and the transformation equation is

$$\vec{e}_{2,l}' = M_2 \vec{e}_{2,l}. \quad (14)$$

Based on the three positional information of the eye, the nose top and the mouth, M_2 is determined and used for the profile normalization. This procedure guarantees corresponding points in the two views to have the same heights.

In order to provide the depth information of the constraint points for the MQVM algorithm, a set of facial features are located manually in the normalized frontal view and profile view. Though a lot of efforts are put into the research of automatic facial feature detection [8,9], the robustness and accuracy are still not satisfactory. The inaccuracy of the feature locating affects feature-based face recognition systems greatly and it's still an open question how to automatically locate the features efficiently and accurately. The

positions of features in the frontal view provide the constraint point set δ in section 2, while the depths of the points are provided by the *x*-coordinates in the profile view. Let (x_f, y_f) and (x_p, y_p) denote the position of a feature in frontal view and profile view respectively. The corresponding constraint point is expressed as

$$c_{(x_f, y_f)} = x_p. \tag{15}$$

Fig. 3 shows the features located on the normalized face views. The white dots are the feature points and the gray lines connect the adjacent features to show their relative positions and correspondence. Since a half of the face is visible in the profile view, it is only possible to locate features on a half face. However, based on the assumption that human face is bilateral symmetric [2], the feature locations can be mirrored to the other half. The MQVM algorithm described in section 3 is then applied on the constraint point set C to approximate a C^2 continuous face 3D surface. Fig. 4 indicates a sequence of MQVM approximations on the face of Fig. 3. The face shapes are under 30° and 90° rotations in depth. The control lattices of the three rows are 16×16, 64×64, and 512×512 which is the finest. It takes 10,000 iterations for each resolution level and totally 80,000 iterations to produce the 3D face surface.

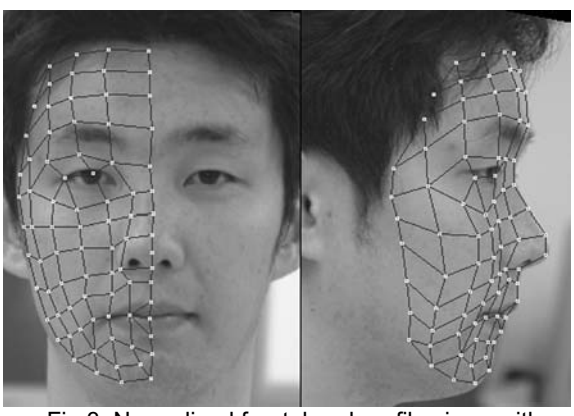

Fig 3. Normalized frontal and profile views with marked features.

5. Virtual view synthesis

We define a plane in 3D space by three adjacent points on the face surface. Then a mesh framework is established for the face model. Since the 3D information of the face is obtained, it is then possible to manipulate the face by 3D rotation, scaling and translation. For post-invariant face recognition, the model is rotated and scaled to generate novel views under different viewing directions. Fig. 5 shows the virtual face views generated by rotating the 3D face surface and texture mapping the frontal face view onto the surface. The 3D face surface is rotated under both tilt and yaw to produce virtual face views under different viewing directions. These face views are stored in a face database and view-based face recognizer can match an input face view under an arbitrary viewpoint against the virtual views and find the best match. In Fig. 6, a comparison between the real profile view of the face and a virtual profile view is made and shown. The profile curve is accurate and the virtual view is realistic.

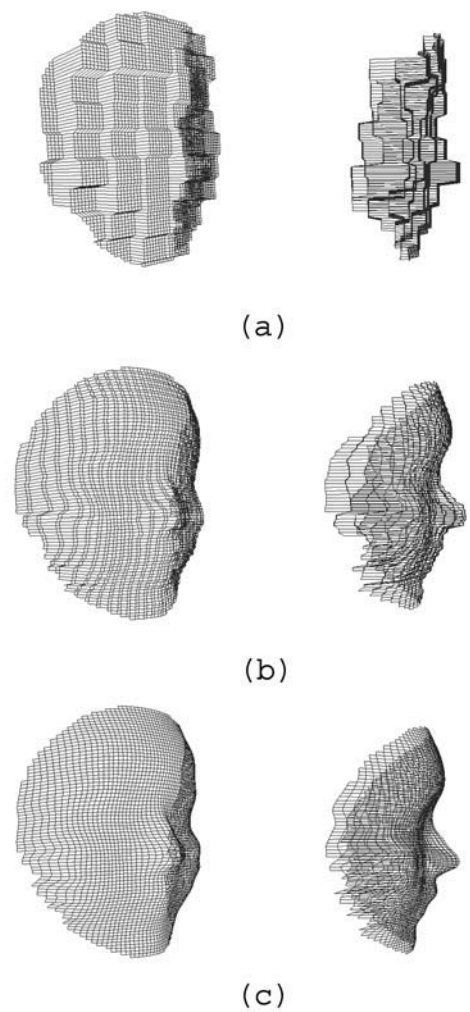

Fig 4. Face surface approximations produced by MQVM algorithm. (a) Face shapes under a 16x16 control lattice, (b) face shapes under a 64x64 control lattice, (c) face shapes under a 512x512 control lattice.

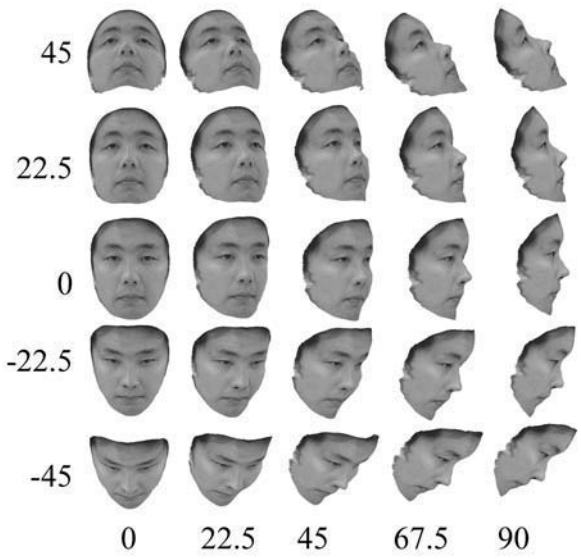

Fig 5. The generated virtual views using the proposed method.

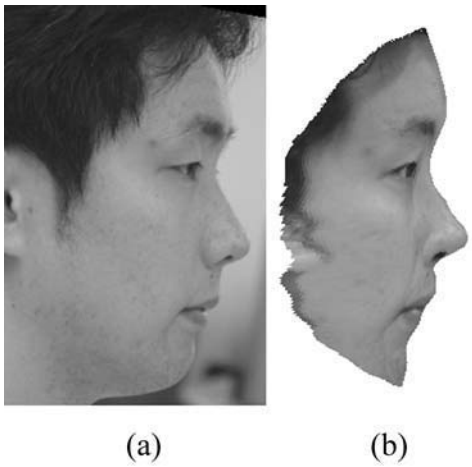

Fig 6. The real profile view (a) and the virtual profile view (b).

6. Conclusion

This paper describes a new algorithm using Multilevel Quadratic Variation Minimization algorithm for constructing 3D individualized face models for the purpose of pose-invariant face recognition. Face views are first normalized, and the facial features are specified on the face views. Then the proposed Multilevel Quadratic Variation Minimization is applied to generate C^2-smooth face surfaces. The algorithm doesn't require generic 3D models or other reference faces and therefore free from the effect of the generic models. It is also more efficient than the single quadratic variation minimization, especially on face surface modeling, where the face images are of high resolution and the facial features are sparse. After the 3D model generation, 2D face views under different viewpoints are synthesized by rotating the 3D model. These virtual views can form a virtual face database for pose-invariant face recognition.

Future work is to design an appropriate texture-mapping algorithm to make the model more realistic.

7. Acknowledgement

This research has been supported by Australian Research Council (ARC) Discovery Grant DP0451091.

8. References

[1] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643-660, 2001.

[2] D. Beymer and T. Poggio, "Face recognition from one example view," *Fifth Int'l Conf. Computer Vision*, pp. 500-507, 1995.

[3] H.H.S. Ip and L. Yin, "Constructing a 3D individualized head model from two orthogonal views," *The Visual Computer, Int'l J. of Computer Graphics*, vol. 12, no. 5, pp. 254-266, 1996.

[4] F. Wallhoff, S. Müller, and G. Rigoll, "Hybrid face recognition systems for profile views using the mugshot database," *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 149-156, 2001.

[5] W. Lee and N. Magnenat-Thalmann, "Fast head modeling for animation," *J. Image and Vision Computing,* vol. 18, no. 4, pp. 355–364, 2000.

[6] D.M. Blackburn, J.M. Bone, and P.J. Phillips, "FRVT 2000 evaluation report," *http://www.frvt2000.com/,* 2001.

[7] Y. Gao, M.K.H. Leung, "Face recognition using line edge map," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 764-779, 2002.

[8] D. Cristinacce, and T.F. Cootes, "A comparison of shape constrained facial feature detectors," *Sixth IEEE Int'l Conf. Automatic Face and Gesture Recognition,* pp. 375-380, 2004.

[9] Z. Xue, S.Z. Li, and E.K. Teoh, "Facial feature extraction and image warping using PCA based statistic model," *2001 Int'l Conf. Image Processing,* vol. 2, pp. 689-692, 2001.

[10] D.G. Luenberger, "Introduction to linear and nonlinear programming," *Addison-Wesley Publishing Company,* 1973.

[11] M.H. Yang, D.J. Kriegman, and N. Ahuja, "Detecting faces in images: a survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34-58, 2002.

[12] W.E.L. Grimson, "A Computational Theory of Visual Surface Interpolation", *Philosophical Trans. Royal Society of London,* Series B, no. 298, pp. 395-427, 1982.