Using 3D Models for Improving Face Recognition

Zoran Bikicki, Ivan Milenković, Dušan Starčević

Faculty of Organizational Sciences, University of Belgrade, Belgrade, Republic of Serbia

zoran.bikicki@mmklab.org, ivan.milenkovic@fon.bg.ac.rs, dusan.starcevic@fon.bg.ac.rs

Contribution to the state of the art

DOI: **10.7251/JIT1402055B** UDC: **004.6.056.55**

Abstract: Face recognition algorithm Principal Component Analysis (PCA) has a significant performance drop when comparing photographs taken from different angle. In this paper a 3D model was used for improving that performance. Model enables us to transform the face image which is taken from certain angle to en face. Model has been tested against biometric database formed at the Faculty of Organizational Sciences. Image rotation based on the model was performed before matching with the en face images from the database. Study results show that algorithm precision on biometric verification and identification has been seriously improved.

Keywords: biometrics, face recognition, 3D graphics, PCA

Introduction

Human face represents a specific characteristic of each human being. When interacting with other people, humans direct attention toward their face. Facial characteristics are used to recognize friends and family members from other, unknown people. Human brain is relatively successful in facial recognition, even under difficult conditions, such as bad lightning, face aging, use of spectacles or haircut change. Although this recognition may seem simple, it is not. It is performed by neural networks located in human brain, which process images collected with eyes, compare them with stored face images and make recognition decision.

Applying computers to this task does not make facial recognition less complex. For example, if we have a collection of photos of certain person, probably the first idea would be to compare pixels or photo histograms. These algorithms are successful when dealing with image search on the internet, or protecting copyrighted material. However, in access control or surveillance system, computer does not work with copies, but with photographs of live humans. They move around, change haircut, wear glasses, and change facial expressions, making recognition more difficult. A significant recognition problem is the fact that actual persons exist in tridimensional space, but they are usually represented by two-dimensional photos.

Automatic facial recognition with use of computers is still in development, with numerous challenges remaining to be solved. In laboratory conditions, without outside interferences, modern face recognition systems have good performances. However, problems such as change in lighting, camera angle and occlusion have a significant impact on biometrics system performance.

Because cameras became even cheaper and widely available, biometric systems for face detection and recognition have numerous applications in different areas. They are used for facial recognition in photographs on social networks, election fraud detection, access control systems and also surveillance systems.

This paper will examine the change of head position on a face image. This problem is especially important for face recognition algorithms which use 2D face images. For evaluation purposes, eigenface algorithm will be used. In order to improve the performance of the algorithm, it is necessary to "draw" the image in three dimensions. With such image we could change its position – turn it until the proper frontal position is reached. Use of frontal images should positively impact precision of eigenface algorithm.

To return face to the frontal position we will use a three-dimensional model of the face that we plotted by using the graphics card. Necessary requirement for successful mapping is to manually install the model in such a position in which certain characteristic points of the face and models coincide. These points will be the central point of the eyes, and the tip of the nose. In this way, we will be able to precisely determine the position of the face.

Face recognition

Face recognition includes several different tasks:

- **Identification –** for a given photo, system tries to find matching person in the database, or to conclude that person on the photo does not have any biometric data in the database
- **Verification –** for each pair <image, person>, system has to verify if biometric data belongs to identified person identity, or not
- **Segmentation –** An array of images is grouped in accordance with persons' identities.

First recognition algorithms were based on facial features. As a general principle, all of these algorithms try to detect key face elements, like eyes, eyebrows, cheeks, nose and other. These data is used to calculate dimension of key face elements, view angle and distances between different parts of the face. Most algorithms use around 40 key elements, and performance of algorithms somewhat varied. Paper [4] describes such algorithm. However, landmark approach has not shown satisfying precision. The main reasons for poor performances are difficulties connected with correct identification of face elements, and neglect of information found in face textures.

The second group of algorithms which arose later, and which still hold primacy in this area are based on the observation of the face as a whole (appearance based). Algorithms from this category store face templates in biometric templates, and match templates as whole with acquired face data. There are numerous algorithms from this category, from which some are frequently used and very popular.

The first algorithm in this group is the so-called Eigenface algorithm [7]. This algorithm is based on the principal component analysis. Principal component analysis is a statistical method for the reduction of a number of variables that are observed in a small number of new variables called principal components. The idea is to find the characteristic vectors (eigenfaces) on the covariance matrix of a face image set, where we treat each image as a vector in a multidimensional space. Graphical representation of eigenfaces is shown in Fig.1. Template of each person in the database is stored as a vector, more precisely a linear combination of these characteristic vectors of the face. For comparing vectors obtained from different persons, we can use different methods, such as Euclidean or Mahalanobis distance.

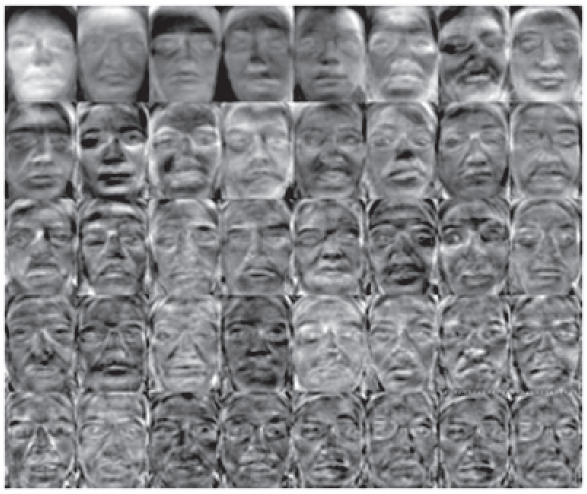

Figure 1. Graphical representation of eigenfaces

Implementation of the eigenface algorithm is availabe within the OpenCV computer vision library. OpenCV is an open source library that contains functions for implementation of computer vision. It is written in the programming languages C and C++, and has support for Windows, Linux and Mac OS X operating systems. OpenCV was used for testing effect of proposed solution on system precision.

3D modelling

Three-dimensional modeling is a set of techniques that are used for creating, modifying and finishing three-dimensional models. Most famous professional tools are “3D studio max” or “Maya”. These tools are very expensive, both costing nearly $ 6,000 for the basic version [10]. There is also some free software in this domain, such as “Blender”. For the purposes of this paper, we have taken an open source model of the head and developed software for modeling, by removing all polygons that do not belong to the face. The result of the modelling is a three-dimensional model, and this work is mainly concerned with artists, architects or designers. In addition, models can be generated by using computers, which require some description and a set of functions.

There are many possible ways to memorize a three-dimensional model. File formats also vary, as models can be textual - that people can easily read, or binary - that is a collection of bytes that computers can easily load to memory.

An example of textual file format is the “obj” [9], in the positions of points are recorded as text. Lines beginning with # character are comment lines, which is common in text files. Symbol “g” marks the group, and “usemtl” describes what texture is to be used. This is followed by points in space (“v” from “vertex”), texture coordinates (“t” from “vertex texture”), normal (“n” from “vertex normal”) and the surface (“f” from “face”). Surfaces are actually triangles that make up the three points in space and it is the most common way of marking the model. Normals are used for lighting, but they can also be omitted and after subsequently calculated. An example of this file can be seen in Table 1. The model can have one or more of these groups, which are called meshes. This file format is simple and readable, but the models remembered in this format take up more disk space and are slower to load into memory, because words need to be converted to numbers.

Table 1. An example of “obj” file

```
# WaveFront *.obj file (generated by CINEMA 4D)
g wheel_3
usemtl _06_kolonki__spec_
v -1.915311 -0.144131 3.745234
v -1.915311 -0.14204 3.640694
v -1.915311 -0.1164 3.639594
vt -22.3438 23.506798 0
vt -19.2136 23.569401 0
vt -19.180599 24.336998 0
f 1/1 2/2 3/3
```

A significant part of each three-dimensional model is the texture. Texture represents the image that is “glued” on certain parts of the model. It is possible to use multiple modes for setting up the textures. In one case, where the coordinates can take values from zero to one, the texture is used as the final image and it is not possible to get out of the frame of the textures. In the second case the texture is repeated, wherein the texture is an infinite plane. One model can use several textures.

Using 3D models for face recognition

As for using 3D models for face recognition there is the paper of Blanz and Vetter - they used a 3D laser scanner for acquiring a database of 200 persons [1]. Afterwards they manually defined points for eyes mouth and nose, and then recognition was being performed by interpolating 3D models from the database. Their research has shown good results.

Other papers are mostly based on face recognition using 3D camera. In this way good results are being given, but this approach calls for expensive equipment. Besides, this cannot be applied to 2D images, so it is not possible to work with existent databases. Some papers using this approach are [3] and [5].

Program description

A program was developed for transforming 2D images so that the face will be en face. It uses an input image as a texture for a head model, rotates the model and exports the rendering.

The program was written in programming language C++. The main reason for selecting that programming language is performance and ability for direct memory access. The 3D graphics library used was OpenGL, which enables graphics hardware acceleration. OpenGL is a low-level API, so programming the code for loading models and camera setup was necessary.

A head model was downloaded from the internet [8]. The model is represented as an array of vertices, array of triangles and array of texture coordinates. Every point in space is an ordered triple (x, y, z), where x, y and z are coordinates of vertex along the suitable axes. Triangles are made up of three vertices and they are usually defined as integers which define the vertex array position because one vertex can be used in many triangles and this way there is no redundant data. Texture coordinate array is the same size as the vertex array, and each vertex has its texture coordinate. Texture coordinate is an ordered pair (u, v), where u and v determine positions on texture along x i y axis.

At the beginning program loads a face image. The image is being rendered on screen and the model is being rendered in front of the image. The bottom-left corner of the image is the origin and the top-right is the point that has x and y axis values equal 1, while z axis (depth) has value zero, as shown in Figure 2. This will enable easier coordinate mapping afterwards. The viewport is being set up as if the camera is positioned in front of the model, looking at the center of the image.

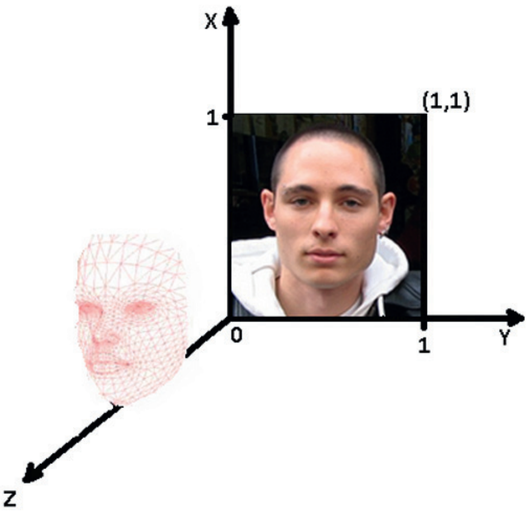

FIGURE 2. IMAGE COORDINATES IN 3D SPACE

Fitting the model and the image

The head model has its scale, while face image has its own. In order to make the model fit the image scale transformation must be applied to the model. Determining the scale will be done by manually selecting three points:

- Center of the left eye
- Center of the right eye
- Peak of the nose

The model will be scaled using the formula:

$$scale = \frac{((\text{imageLeftEyeY} + \text{imageRightEyeY})/2 - \text{noseX})}{((\text{modelLeftEyeY} + \text{modelRightEyeY})/2 - \text{noseY})}$$

The next step is rotating the model so that it is in the same pose as the person on the image. The first step is determining in what direction the head is being rotated in. For that we can use the fact that when a head is being rotated, the tip of the nose moves more than the two eyes, as shown in Figure 3.

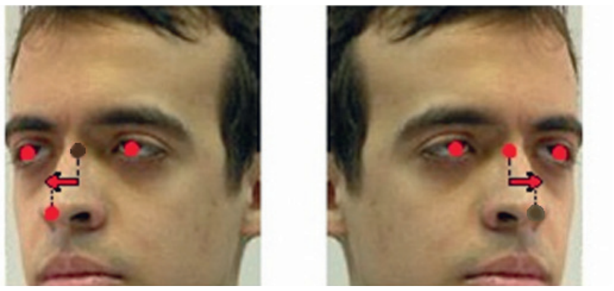

FIGURE 3. DETERMINING THE FACE ROTATION DIRECTION

Approximate rotation along Y axis is determined using the formula:

$$rotY = rotD \times \frac{90}{PI} \times \text{acos}(\frac{\text{imageLeftEyeX} + \text{imageRightEyeX}}{\text{modelLeftEyeX} + \text{modelRightEyeX}})$$

Variable RotD represents rotation direction in equation.

OpenGL uses degrees for rotation, so rotation angle in the formula is being represented in degrees. After rotation, the model is being translated so that all the points have equal coordinates. Afterwards the head model is being manually fine tuned in order to fit the image as best as possible.

Determining texture coordinates

In order to map the texture to the model it is necessary to determine the texture coordinates for each vertex. Because the model is fitted on the image and the image has been placed so that all its pixels fall within the range 0-1, texture coordinates will be determined by projecting the model on XY plane. This is done by multiplying the positional vector of a vertex with a projection matrix. Multiplying is done in homogenous coordinates, which means adding a scale factor as a fourth dimension of a vector. After multiplying, vector has to be normalized so that the fourth parameter has a value of one. A demonstration of a simple translation transformation is being shown in Figure 4. Projection matrix will be determined using OpenGL function which returns model view matrix: glGetFloatv with parameter GL_MODELVIEW_MATRIX.

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ z + dz \\ 1 \end{bmatrix}$$

Figure 4. Projection matrix

Hidden surfaces

When one looks at an image of a 3D object, one cannot see all its parts. If the object is a box, only its three sides out of six are visible at a given moment. The sides that are not visible are called hidden surfaces. Checking if surface is hidden is done by checking vertex order when projected on a 2D plane - surface is hidden if vertices are ordered counter-clockwise. Hidden surfaces cannot be mapped on a model, so we will use a mirror function to fill in those gaps. The model is symmetrical so we will determine a mirror vertex for each vertex in the model. While rendering the model, hidden surfaces will be rendered using mirror vertices.

Saving transformed image

After determining texture coordinates and hidden surfaces, all that is left is to return the model to its original position - facing camera, rendering it and saving the pixel data to a file for later use. While fitting the model we used only OpenGL transformations for projection, so returning the model to its original position consists only of resetting OpenGL model view matrix to identity matrix. Pixel data will be fetched using OpenGL function glReadPixels, and data will be saved in bmp file format which uses the same data format as graphics cards**.**

Program usage

The program shows the first image in database and the head model. User selects the three points: center of the left eye, center of the right eye and tip of the nose and the program fits the model approximately. If the result is not appropriate user can select the three points again. After that, the model is fitted additionally using keyboard. If the model is too wide or narrow it is possible to scale the model along X axis. The user can change model transparency at any time in order to see its position better. When fitted, pressing return button the texture is being mapped to the model. Once mapped, the user can turn the wire overlay on or off and rotate the model. After that the model is being brought back to en face position and newly made image is being saved to the hard disk. This process is shown in Figure 5. The loss of quality can be observed on the result image, on parts where mirror mapping is applied. This can be attributed to non-uniform lighting of the original image.

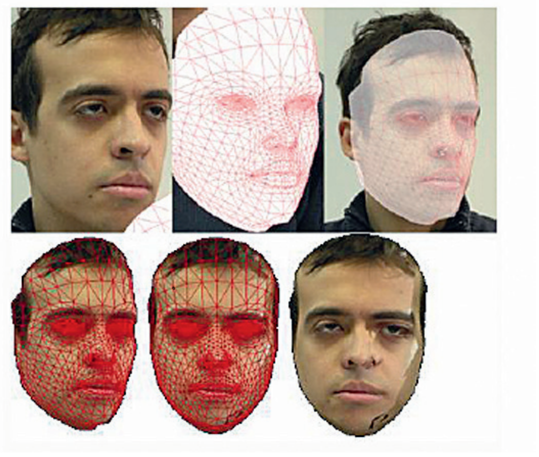

Figure 5. Model fitting, rotating and rendering

Performance evaluation

All tests were performed on face database collected at the Faculty of organizational sciences [6]. Database consists of biometric data collected from 34 different persons. First collection of match scores was obtained by matching a base set of en face oriented photos with a set collected from same persons, in a similar way, six months later. Second collection of match scores is result from comparing base set with a set of images normalized by our program. Overall, we have done three types of comparisons:

- matching base set with en face images from later period
- matching base set with half profile images from later period
- matching base set with images normalized by our program (transformed from profile angle to en face)

Calculated match scores were used to draw ROC („Receiver Operating Characteristics“) curves, which allows us to visualize results. Figure 6 shows part of the curve on which the FMR (False Match Rate) value is lesser than 5 percent. Identification rate was also calculated.

Calculating identification rates leads us to following conclusions:

As expected, en face images have shown best performance. Twenty two persons were successfully identified (65%).

Because of changed angle of view, half profile images have shown significant performance drop. Only 13 persons were successfully identified (38%).

Normalized images have performed better than half profile images, but worse than en face. Twenty one person has been successfully identified, just one person less than enface images.

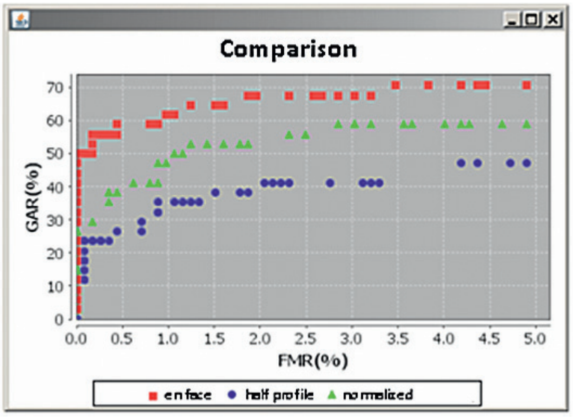

FIGURE 6. FACE RECOGNITION RESULTS

These results are encouraging since the improvement of face recognition algorithm is significant. On the other hand, it can be observed that the transformation is not perfect, since the results are not the same as original en face images comparison. This may be happening because head model is not flexible, or a consequence of loss of data from the side of the face that is turned away from the camera.

CONCLUSIONS

In this paper computer graphics techniques were used to improve face recognition performance. To evaluate proposed techniques, an application was developed. It allows us to transform two-dimensional photos to three-dimensional models. Three-dimensional model is then rotated in order to transform face images from profile to en face position. In this way, problems with camera angle are at least partially solved. Evaluation was performed on facial database collected at the Faculty of organizational sciences, and results have shown better precision after normalization of the photos.

For now, the process of model fitting is done manually, which means that human presence is necessary in order for system to function. Subsequent versions of the program should independently determine the position and angle of the head, which would automate the system. This automation would enable more intensive testing on larger databases.

Face model used for texture mapping is somewhat rigid and it is not flexible. It would be appropriate if all people had the same face shape, which is not the case. Face model should be fitted in accordance with facial features of an individual. Face specific parameters could be used for recognition, independently or together with other algorithms. Also, application should be extended to allow illumination normalization.

Acknowledgment
This work is a part of the project Multimodal biometry in identity management, funded by the Ministry of Education and Science of Republic of Serbia, contract number TR-32013.

References

[1] Blanz, V. & Vetter T. (2003). *Face recognition based on fitting a 3D morphable model*, Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.25, no.9, pp.1063,1074.

[2] Bradski, G. & Kaehler, A. (2008). *Learning Opencv, 1st Edition (First ed.)*, O'Reilly Media, Inc.

[3] Bronstein, A., Bronstein, M. & Kimmel, R. (2005). *Three-Dimensional Face Recognition*, International Journal of Computer Vision, vol. 34.

[4] Brunelli, R. & Poggio, T. (1993). *Face recognition: Features versus templates*, IEEE Trans. Pattern Anal. Mach. Intell.15(10), pp. 1042–1052.

[5] Kakadiaris, I., Passalis, G., Toderici, G. & Murtuza, M. & Lu, Y. & Karampatziakis, N. & Theoharis, T. (2007) "*Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach*", IEEE Trans. Pattern Anal. Mach. Intell.29(4), pp. 640–649

[6] Milenković, I. & et al. (2011). *A multimodal biometrics system implemented using open source technology*, Telecommunications Forum (TELFOR), 2011 19th , pp.1352 - 1355.

[7] Turk, M. & Pentland, A. (1991). *Eigenfaces for recognition*, Journal of Cognitive Neuroscience 3(1), pp. 71–86.

[8] Female head by Bitmapworld http://www.turbosquid.com/3d-models/polygonal-female-head-3ds/265937, last visited on 10th of June, 2013.

[9] Obj file format description, http://www.cs.cmu.edu/~mbz/personal/graphics/obj.html , last visited 15.11.2014.

[10] Official 3dmax website, http://www.autodesk.com/products/3ds-max/buy, last visited 14.11.2014.

Submitted: December 2, 2014.
Accepted: December 6, 2014.