Rapid Implementation of 3D Facial Reconstruction from a Single Image on an Android Mobile Device

Phuc Huu Truong[1], Chang-Woo Park[1], Minsik Lee[2], Sang-Il Choi[3], Sang-Hoon Ji[4] and Gu-Min Jeong[1*]

[1] School of Electronics Engineering, Kookmin University
Seoul, Korea
[e-mail: gm1004@kookmin.ac.kr]
[2] School of Electrical Engineering and Computer Science, Seoul National University
Seoul, Korea
[3] Department of Applied Computer Engineering, Dankook University
Gyeonggi, Korea
[4] Department of of Applied Robot Technology, Korea Institute of Industrial Technology
Ansan, Korea
*Corresponding author: Gu-Min Jeong

Received December 23, 2013; revised March 14, 2014; accepted April 10, 2014; published May 29, 2014

Abstract

In this paper, we propose the rapid implementation of a 3-dimensional (3D) facial reconstruction from a single frontal face image and introduce a design for its application on a mobile device. The proposed system can effectively reconstruct human faces in 3D using an approach robust to lighting conditions, and a fast method based on a Canonical Correlation Analysis (CCA) algorithm to estimate the depth. The reconstruction system is built by first creating 3D facial mapping from a personal identity vector of a face image. This mapping is then applied to real-world images captured with a built-in camera on a mobile device to form the corresponding 3D depth information. Finally, the facial texture from the face image is extracted and added to the reconstruction results. Experiments with an Android phone show that the implementation of this system as an Android application performs well. The advantage of the proposed method is an easy 3D reconstruction of almost all facial images captured in the real world with a fast computation. This has been clearly demonstrated in the Android application, which requires only a short time to reconstruct the 3D depth map.

Keywords: 3D facial reconstruction, Depth map estimation, Facial recovery, Three-dimensional display, Smartphone

This work was supported in part by Global Scholarship Program for Foreign Graduate Students at Kookmin University in Korea and also supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2011-0006958).

http://dx.doi.org/10.3837/tiis.2014.05.011

1. Introduction

3D facial reconstruction is a useful research discipline because of its wide range of application, from face recognition and 3D animation to video conferencing. A 3D model reconstruction of a human face can be applied to an animation using a model [1] [2] or video-based [3]. Moreover, this facial recognition can also be utilized to recognize a human face by extracting the 3D geometry information of the face and generating virtual samples by rotating the resulting 3D face model [4] [5] [6].

To date, many techniques and algorithms have been proposed to solve this problem. Nevertheless, most of the researches handle only one or some of the parameters, and the results remain unsatisfactory. To estimate a pose parameter, Choi et al. [7] proposed a method using an Expectation Maximization (EM) algorithm based on a weak perspective projection by calculating the sum of the posterior probabilities of all the 3D feature points. A combination of the EM algorithm and a 3D face shape model was suggested in [8] and [9] to accommodate additional poses. To determine the surface characteristics, methods using stereo images were proposed in [10] and [11]. The stereo vision method infers information regarding a 3D structure and the distance of a scene from two or more images taken from two or more cameras from various viewpoints. In [10] and [11], it is necessary to first find the correspondences of each image pixel from the different cameras. A 3D structure is then built from these correspondences. The correspondences include intrinsic and extrinsic parameters. An intrinsic parameter is the mapping of an image point from one camera to the pixel coordinates in all other cameras. An extrinsic parameter describes the relative position and orientation of the images that are built based upon the rotation, translation, and scale. A simple and effective approach exists for 3D reconstruction based on statistical models of objects [2][5][6][12][13]. This model-based approach imposes model constraints to liminate any ambiguity and guarantee a unique solution. Good results from 3D reconstruction have been realized based on the statistical model learned from training samples. The advantage of this model-based approach is that only one single image is used to realize a reconstruction.

Nowadays, the smartphone market is expanding rapidly. It is not only the fundamental applications but also the computational applications related to research that are developed and ported to the smartphone because of its convenience, mobility, and popularity. In this technology trend, the 3D reconstruction discipline, especially, the reconstruction of the 3D human face, is essential to mobile devices. By porting to a smartphone, the algorithms for 3D reconstruction can be tested conveniently and in a less-costly manner in the real world instead of a complex and expensive system in the past. Lee et al. [14] created a photorealistic 3D face model on mobile devices using an active contour model (ACM). A generic 3D mesh model is first deformed, the features of a human face are extracted from front and profile images, and an interpolation is used to add them into the model. However, this technique requires two images and a generic 3D model to reconstruct the 3D face. Wang et al. [15] used a photometric stereo algorithm to reconstruct a 3D model on Android phones from four images of an object captured from a phone's built-in camera. This technique can be used to reconstruct a 3D model quickly; however, is complex, as it requires four images under various lighting conditions.

In this paper, we propose a rapid 3D facial reconstruction implementation using a single frontal image on a mobile device. We do not impose strict camera conditions, such as illumination or camera calibration. Some optimizations are integrated to increase processing ability and improve the runtime execution of the program. The proposed implementation

works well with an arbitrary illumination condition of a realistic image captured in the real world with a camera. The remainder of this paper is organized as follows. Section 2 provides background knowledge and the algorithms related to face detection and 3D facial reconstruction. Section 3 describes the proposed 3D facial reconstruction system. In Section 4, an application designed for use on mobile devices is presented. The performance of the application on an Android device is illustrated in Section 5. Finally, Section 6 provides some concluding remarks regarding the proposed implementation.

2. Preliminaries

This section provides a summary of the theories, and their corresponding mathematical equations related to the system proposed in this paper. To indicate the relation with the proposed implementation, and for a clear description of the background of this work, we have arranged the following sections corresponding to the sequence of 3D facial reconstruction.

2.1 Face Detection

In this paper, we use the Adaboost method, proposed by Viola and Jones [20], to detect human faces. The principle idea of this method is to combine many weak classifiers to create one strong classifier. Four features were used in the adoption of the framework. These features are based on Haar wavelets. However, they use rectangle combinations instead of true Haar wavelets, and are therefore called Haar-like features. **Fig. 1** illustrates some examples of these features.

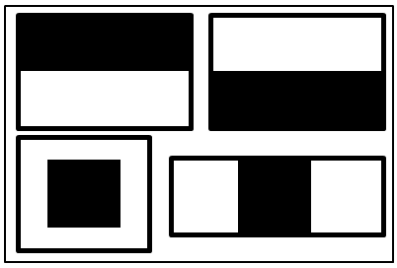


Fig. 1. Haar-like features

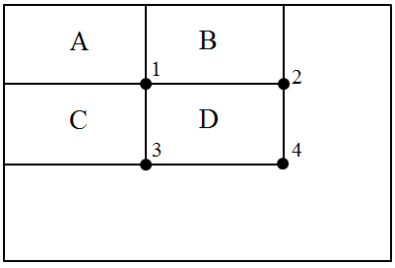


Fig. 2. Integral image of a rectangle

To determine the presence of Haar-like features, an integral image technique that calculates the integral value of a pixel by summing all the values of the pixels above it and on the left is used. For example, the sum of the pixels within rectangle D in **Fig. 2** can be computed as $4+1-(2+3)$, or as (x4, y4) + (x1, y1) - (x2, y2) - (x3, y3). Adaboost is used to select the features and train the classifiers. A series of Adaboost classifiers is combined into a filter chain to classify an image. A sub-window with a fixed size of 24x24 pixels is extracted in one picture, and the corresponding Haar-like features are calculated. The sub-image passes each filter sequentially via comparison with the acceptance threshold in each filter. Sub-images that pass through all the filters are determined to be face images. The order of the filters in a cascade is based on the weight assigned by Adaboost during the training step.

2.2 Shape recovery from face image

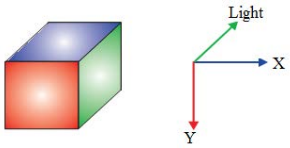
It can be assumed that the apparent brightness of human faces to an observer is the same regardless of the observer's angle of view [12]. In other words, we can consider a human face

as an ideal diffused surface, or a Lambertian surface [18] [19]. The face image can therefore be approximated as a linear equation:

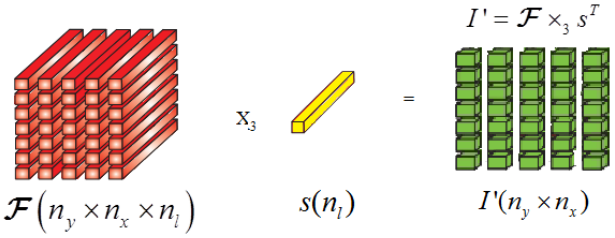
$$I(x,y) \approx f(x,y)^T s.$$

To represent a face model as a tensor, we reformulate it into a tensor equation:

$$I' = \mathbf{F} \times_3 s^T.$$



a) Illustration of the surface



b) Illustration of the tensor multiplication

Fig. 3. Illustration of the face model in tensor

Fig. 3 illustrates the surface tensor $\mathbf{F}$ and a multiplication of this tensor with a lighting condition s. A proper representation of ϕ exists such that $\mathbf{F}$ is a linear function of ϕ as in the following:

$$\mathbf{F}(\phi) = \overline{\mathbf{F}} + \mathbf{T} \times_4 \phi^T.$$

Here, $\overline{\mathbf{F}}$ and $\mathbf{T}$, which are the average surface tensor and the reduced-dimension version of the spherical harmonic tensor, respectively, are calculated using N-mode SVD for training samples. The detailed explanation of this method can be found in references [12] and [13]. The image can thus be described as a function of the identity vector ϕ and light condition s.

$$I(\phi, s) = \left(\overline{\mathbf{F}} + \mathbf{T} \times_4 \phi^T\right) \times_3 s^T.$$

2.3 Alternating Least Squares (ALS) Optimization

For each new image of a person, we must find the pair of ϕ and s that satisfies:

$$I' = \left(\overline{\mathbf{F}} + \mathbf{T} \times_4 \phi^T\right) \times_3 s^T. \quad (1)$$

This work can be considered as finding such a pair in a nonlinear least-squares optimization by reformulating (1) as follows:

$$\text{minimize} \qquad \|K\|^2 = \left\| I' - \left(\overline{\mathbf{F}} + \mathbf{T} \times_4 \phi^T\right) \times_3 s^T \right\|^2, \quad (2)$$

and applying N-mode SVD operation

$$\mathbf{T} = \mathbf{C} \times_1 U_y \times_2 U_x \times_3 U_l .$$

Here, U_y, U_x, U_l are unit vectors calculated from Mode-1, Mode-2 and Mode-3 SVD operators, respectively, and $\mathbf{C}$ is the core of the N-Mode SVD operator.

$$\Rightarrow \|K\|^2 = \left\| I' - \left(\overline{\mathbf{F}} + \mathbf{S} \times_1 \mathrm{U}_y \times_2 \mathrm{U}_x \times_4 \phi^T \right) \times_3 s^T \right\|^2 , \tag{3}$$

$$\text{with } \mathrm{S} = \mathbf{C} \times_3 \mathrm{U}_l = \mathbf{T} \times_1 \mathrm{U}_y^T \times_2 \mathrm{U}_x^T .$$

According to the demonstration of Lee. et al. [12], the value of $\|J\|^2$ concentrates dominantly in the matrix space $\left(U_y^T, U_x^T\right)$. Therefore, (3) can be simplified into a new problem:

$$\text{minimize} \qquad \|J\|^2 = \left\| \mathrm{L} - \left(\mathbf{R} + \mathbf{S} \times_4 \phi^T \right) \times_3 s^T \right\|^2 . \tag{4}$$

Here, $$\mathrm{L} = I' \times_1 \mathrm{U}_y^T \times_2 \mathrm{U}_x^T , \mathbf{R} = \overline{\mathbf{F}} \times_1 \mathrm{U}_y^T \times_2 \mathrm{U}_x^T .$$

To determine an optimal solution to ϕ and s, we apply the ALS method. We first optimize (4) with respect to s for a fixed ϕ, and then with respect to ϕ for a fixed s. This process is then repeated until a convergence is achieved. Optimizing with respect to either ϕ or s provides a linear least squares problem and the solution can be found easily in a closed form. First, we set $\phi \leftarrow 0$ (mean face), and then calculate the following equations iteratively,

$$\begin{aligned} \mathcal{V} &\leftarrow \mathbf{R} + \mathbf{S} \times_4 \phi^T , \\ s &\leftarrow \mathcal{V}_{(3)}^{+T} \mathrm{l} , \\ \mathcal{W} &\leftarrow \mathrm{S} \times_3 s^T , \\ \phi &\leftarrow \mathcal{W}_{(3)}^{+T} \left(\mathrm{l} - \mathbf{R}_{(3)}^T s \right) . \end{aligned}$$

Here, $(.)^+$ denotes the Moore-Penrose pseudo-inverse and l is the vectorized version of L. This iteration is executed until the norm of the change of s is less than a predefined threshold ε.

2.4 Canonical Correlation Analysis (CCA) Algorithm

It is known that not all component variables in the parameter vector have the same contribution to the mapping task, and that redundancy and noise exists among these variables may have a negative consequence on the mapping. Therefore, we first apply the CCA approach on two spaces to find the most correlative and complementary factors and then build the mapping based on these factors.

A CCA is a very powerful tool for finding the linear relationship between two sets of multi-variate measurements in their leading factor subspaces. Similar to a principal components analysis (PCA), a CCA also reduces the dimension of the original sets because only certain pair data are required to estimate the relationship between two sets. Nevertheless, a CCA can deal well with two multidimensional spaces, and it is therefore much better at the regression than a PCA.

Consider the linear combinations $X = [x_1, x_2, ..., x_n]$ and $Y = [y_1, y_2, ..., y_n]$ of two variables. Finding the canonical correlation between these sets can be considered as finding the optimal linear projective matrices, $W^x = \left[w_1^x, w_2^x, ..., w_d^x \right]$ and $W^y = \left[w_1^y, w_2^y, ..., w_d^y \right]$, also called

canonical projection pairs, such that $x_i' = X^T w_i^x$ and $y_i' = Y^T w_i^y$ are the most correlated. This can be achieved by maximizing the following correlation:

$$\rho\left(w_i^x, w_i^y\right) = \frac{E\left[x_i'^T y_i'\right]}{\sqrt{E\left[x_i'^2\right]E\left[y_i'^2\right]}} = \frac{w_i^{xT} C_{xy} w_i^y}{\sqrt{w_i^{xT} C_{xx} w_i^x w_i^{yT} C_{yy} w_i^y}},$$

$$\text{subject to } \rho\left(w_x^j, w_y^i\right) = \rho\left(w_x^i, w_y^j\right) = 0$$

$$\text{for } j = 1, 2, ..., i-1,$$

where C_{xx} and C_{yy} are the within-set covariance matrices of X and Y, respectively, while C_{xy} denotes their between-set covariance matrix.

Let $A = \begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix}, B = \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{pmatrix}$, the solution $W = \left(W^{xT}, W^{yT}\right)^T$ amounts to the extremum points of the Rayleigh quotient:

$$r = \frac{W^T A W}{W^T B W}$$

The solution W^x and W^y can be obtained by solving the generalized Eigen-problem:

$$AW = BW\Lambda.$$

2.5 Feature mapping

Model-based mapping is used to calculate the depth information. In this work, a CCA is utilized to find the mapping between the depth and face surface. M. Reiter [13] suggested predicting the depth maps from an RGB face image utilizing a CCA for a set of pairs of RGB image data vectors and corresponding depth maps. This method can be easy to use; however, the accuracy of the result is diminished because it is difficult to find an accurate and direct map between the image data and corresponding depth. M. Lee et al. [12] estimated a depth map from a personal identity vector by utilizing a CCA to find the correlation, i.e., a linear relationship between personal identity vectors transformed from the normal vectors of a face and their corresponding depth, d. This is a fast and easy technique for estimating depth. The experimental results in the reference [12] show the effect of this technique in terms of depth estimation.

Therefore, in this paper, we apply this CCA-based technique in the modeling step. The procedure used in this work is as follows.

Step 1: Estimate the leading factor pairs $W^x = \left(w_1^x, w_2^x, ..., w_k^x\right)$ and $W^y = \left(w_1^y, w_2^y, ..., w_k^y\right)$ from samples of N-pairs of identity vectors and depth tensors.

Step 2: Compute the regression parameter matrix.

Step 3: Identify the linear mapping between the identity vector and the depth map.

This method is simple to use, and its computation ability is efficient to apply in a smartphone. The personal identity vector of each face image ϕ is calculated from optimizing the nonlinear equation (4) using the Alternating Least Square (ALS) technique. The correlation parameter is built in the modeling step by applying the CCA for the personal

identity vector, ϕ, of each image and its corresponding depth map, d, in training samples. The procedure of modeling is described in **Fig. 3**. In the reconstructing step, the identity vector of the testing image is used to reconstruct the depth information based on the correlation parameters trained in the modeling step by the CCA algorithm.

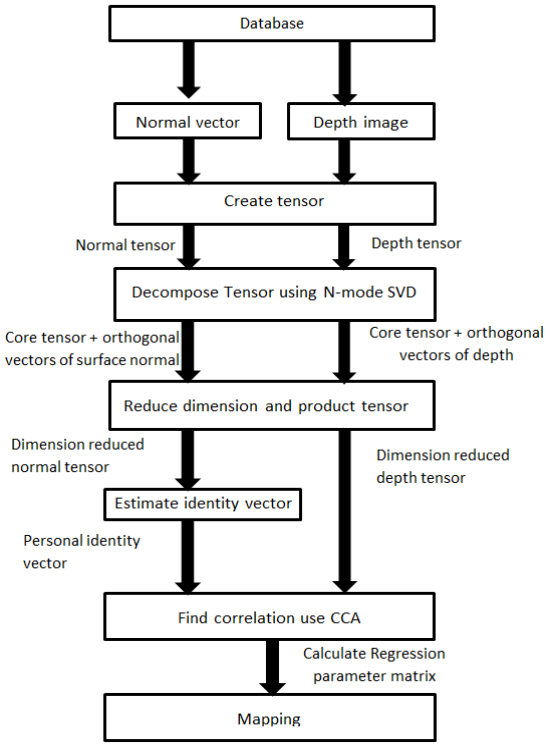


Fig. 4. CCA-based modeling method used on a computer

3. Construction of 3D facial reconstruction system

3.1 Overview of 3D reconstruction from a 2-dimensional (2D) face

The objective of the proposed implementation is to create a system that can form a 3D face from a frontal face image. The system is built on an Android mobile device with the input being a built-in camera; the output, an Android display; and the process object, a face image. The system should automatically detect the object upon input, and acquire the necessary features of the object, specifically the shape of the face. The system then processes these features using the loaded internal parameters to calculate the desire information of the object, that is, the depth image including texture.

To construct the internal parameters of the system, we used the FRGC 2.0 database [16] to train the 3D face modeling. The FRGC database consists of 50,000 recordings captured under varying illumination conditions and in two expressions (neural and smiling). The 3D data are taken under controlled illumination conditions. We select 500 range-data and their corresponding frontal face image from the database. We used 400 samples for training the model that maps between the identity vector and the depth image, and 100 samples to test the

reconstruction on a computer. The experimental results provide a 2.34 voxel mean absolute error of reconstructed depths.

Based on the model, we created an application for 3D human face reconstruction for Android OS-based mobile devices using the results of the modeling implemented on a desktop computer. Because the Android application uses the same algorithms tested in Matlab, its accuracy is the same as that when tested on a desktop computer using Matlab. On the other hand, this application was built for reconstructing 3D faces from images captured in the real life on a mobile device, and therefore real 3D information does not always exist to confirm the accuracy of the result. Therefore, we did not evaluate the accuracy of the application in our approach, although details of the experimental accuracy can be found in the reference [12]. Instead, we measured and evaluated the speed of the application. This is an important factor for its applicability in real life. A short reconstruction will be of service to the development of a real-time implementation as a future work.

3.2 Design of a 3D reconstruction from a 2D face

The system can form a 3D face model on a mobile device by loading a database and executing the modeling process on the device. However, it is inconvenient to have to run the modeling, before utilizing the reconstruction function, every time the system is initiated. Furthermore, mobile devices do not have adequate memory or sufficiently fast and powerful processors to handle such an extremely large 3D database. Therefore, we elected to divide this work into two parts. The first part is the modeling, processed on a desktop computer using Matlab to exploit its computation and memory-access ability to address the many large-memory required tensors. The second part is the application built on Android mobile devices. It reuses the results of the modeling as internal parameters for the 3D reconstruction by loading the descriptor. The modeling results are stored as binary format files in the memory of the mobile device, such as the internal memory or an external memory SD card. We decided to use binary format because it requires smaller memory compared with other format types such as xml or text files. It is also faster to process. The information in these binary format files is loaded and reorganized into tensors with defined orders. The tensors are parameters that are actually used for the 3D reconstruction system.

Fig. 5 describes the architecture of the 3D facial reconstruction system on a mobile device. The face detector is called to identify and crop a frontal human face in an image. A color cropped face is used to extract the texture features that add to the depth map results to recover a full 3D face. Meanwhile, a grayscale cropped face is the main object for reconstructing the depth of each corresponding face pixel.

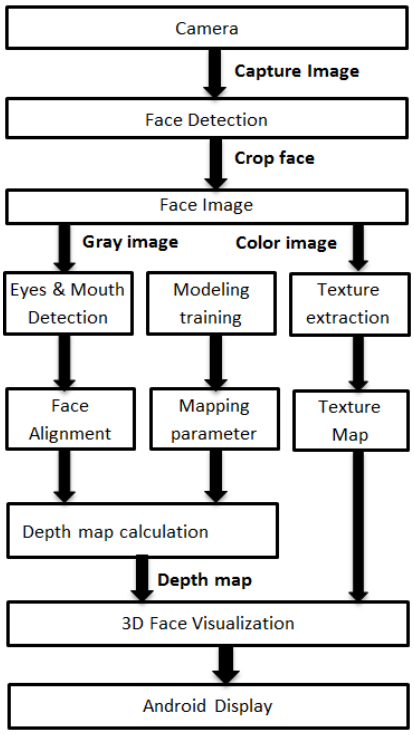


Fig. 5. Architecture of the 3D facial reconstruction application

4. Implementation of 3D facial reconstruction in a mobile device

4.1 Process of 3D facial reconstruction

In this paper, we propose an implementation for automatically reconstructing a 3D image from a single frontal image on an Android-based mobile device using the mapping results from a computer using Matlab. The system will automatically detect, crop, and convert a head-on face in an image taken by a built-in camera of a mobile device into a depth map. The texture of the color cropped face image is extracted and added to the depth map to provide a more realistic look. **Fig. 6** shows the processing modules of the implementation.

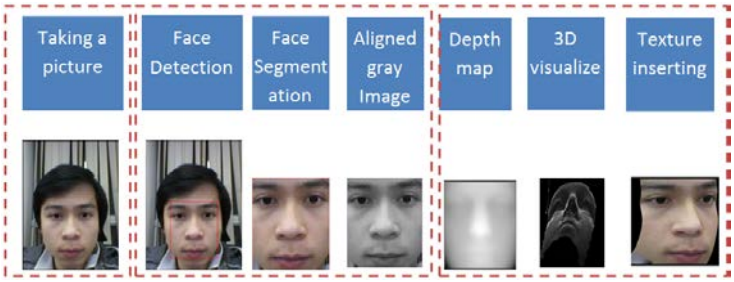


Fig. 6. Sequential processing modules of 3D reconstruction

In the proposed implementation, a picture is taken from the built-in camera of a mobile device. The application can also be tested by loading a sample image store in the phone memory, either the internal memory or an external card. The process of the image is conducted sequentially as follows.

Step 1: A picture is taken or an image is loaded, and the image data are then sent to the face detector to find and crop the frontal human face.

Step 2: The cropped face is affine transformed for normalization.

Step 3: The specific personal identity vector is then calculated from the image using the ALS algorithm.

Step 4: The depth information of the image is reconstructed.

Step 5: Textures from the color cropped face are extracted and inserted into the depth image to obtain the full 3D face.

4.2 Face Detection and Alignment

The face detection and alignment module is a key factor in the application. This module is a verifying gate confirming if an image satisfies the conditions of the applied reconstruction algorithm, i.e., the input image is a frontal face image. This module is also a modifier that standardizes an input image into a suitable form. Specifically, this module applies a Haar-like cascade technique to detect a frontal face existing in an image. If the module can find a face image, the captured image fulfills the input requirement; otherwise, the process stops. To accelerate the face detection on a mobile device, we chose to utilize the OpenCV library. This contains many open-source programming functions mainly aimed at real-time computer vision. It also includes the implementation of a face detection algorithm [21]. Therefore, we utilize OpenCV as a tool to implement the face detection in this module. Face detection with OpenCV support is conducted by calling a Haar-classifier that is loaded from an xml-format file trained to detect frontal faces in images. **Fig. 7** illustrates the face detection and cropping of the application.

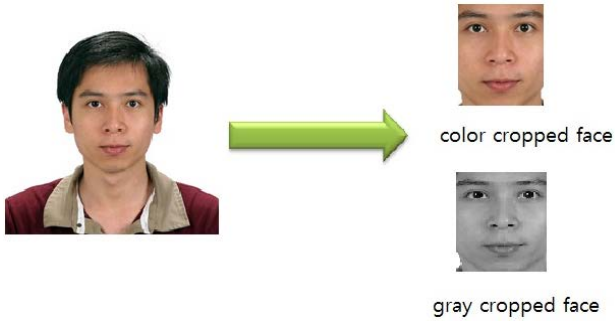


Fig. 7. Face detection and cropping

Because there are morphological differences between different faces owing to human characteristics, such as the length of the face or the distance between the eyes, it is necessary to geographically normalize all the images. To resolve this geographical problem, we apply a landmark-based method. The positions of the eyes and mouth are used as landmarks to represent a facial morphography. We designate a set of landmark positions and use an affine transformation (A,t) to convert the images, I, and spherical harmonic images, Q_i, into a standard form in which the eyes and mouth are in the designated positions.

To detect the eyes and mouth of each cropped face, we reuse the Haar-like cascade technique mentioned in Section 2.1. Three classifiers that are formed from a cascade of Haar-like classifiers are created and trained with the left eye, right eye, and mouth. These three

classifiers are applied to the cropped image to locate the position of the three feature points. Based on this location, the $(\mathrm{A,t})$ transformation can perform properly.

Suppose that $I'(x,y)$ and $Q'_i(x,y)$ are an image and its spherical harmonic image, respectively. An affine transformation is used to transform

$$\begin{Bmatrix} I(x,y) \\ Q_i(x,y) \end{Bmatrix} \rightarrow \begin{Bmatrix} I'(x,y) \\ Q'_i(x,y) \end{Bmatrix},$$

such that

$$\begin{cases} I(x,y) = I(x',y') \\ Q_i(x,y) = Q_i(x',y') \end{cases}$$

with

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + t.$$

Here, $I'(x,y)$ and $Q'_i(x,y)$ are an affine transformed face image and a spherical harmonic image, respectively. $A \in \mathbf{R}^2$ and $t \in \mathbf{R}^2$ are the mapping matrix and translation vector of the affine transformation. Denoting that $A = \begin{pmatrix} a & b \\ b & d \end{pmatrix}$ and $t = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$, we have:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}.$$

Rewriting this equation in the homogeneous form,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix},$$

$$\Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Assuming that (x_l, y_l), (x_r, y_r), (x_m, y_m) are the coordinates of the left eye, right eye and mouth in the original coordinate systems and (x'_l, y'_l), (x'_r, y'_r), (x'_m, y'_m) are their corresponding coordinates in the transformed coordinate systems, A and t are then calculated as follows:

$$\begin{bmatrix} x'_l & x'_r & x'_m \\ y'_l & y'_r & y'_m \\ 1 & 1 & 1 \end{bmatrix} = \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix} \begin{bmatrix} x_l & x_r & x_m \\ y_l & y_r & y_m \\ 1 & 1 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix} = \begin{bmatrix} x'_l & x'_r & x'_m \\ y'_l & y'_r & y'_m \\ 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} x_l & x_r & x_m \\ y_l & y_r & y_m \\ 1 & 1 & 1 \end{bmatrix}^{-1}$$

$$\Rightarrow \begin{pmatrix} A & t \end{pmatrix} = \begin{bmatrix} x'_l & x'_r & x'_m \\ y'_l & y'_r & y'_m \end{bmatrix} \cdot \begin{bmatrix} x_l & x_r & x_m \\ y_l & y_r & y_m \\ 1 & 1 & 1 \end{bmatrix}^{-1}.$$

This transformation ensures that landmark points of all normal and spherical harmonic images are made in the same positions, thereby basically assisting with the image normalization. After cropping and alignment, the image is normalized to 120x100 pixels to estimate the corresponding depth map in the reconstruction step. The normalization leads to an accurate correlation between normal and spherical harmonic images, and improves the accuracy of the reconstruction results. This reduces the complexity of the preprocessing required for the 3D reconstruction.

4.3 3D facial reconstruction

The 3D facial reconstruction module is the core of this application. This module is the engine of the reconstruction because it includes most of the important functionality related to the 3D reconstruction process.

The image is first normalized to improve its effectiveness in determining the personal identity vector.

$$I_{norm}(x,y) = \frac{I(x,y)}{\sqrt{\sum_{x,y} I(x,y)^2}}. \tag{5}$$

This normalized image is the basic input for applying the ALS algorithm to determine the identity vector and lighting conditions of the image. However, to improve the speed of optimization and reduce the computational costs, we utilized the results of an N-mode SVD during the modeling step to reduce the number of dimensions of the image. Details of this can be found in [12]. It should be noted that a reduction in the spatial dimension leads to a reduction in the number of iterative loop of the optimization algorithm.

We denote a reduced image as $L = U_y^T I_{norm} U_x$, and its vectorized version as l. As stated in Sections 2.3 and 2.4, the personal identity vector, ϕ, and lighting conditions, s, of the reduced image are estimated by minimizing the value of $\|J\|^2 = \left\|L - (R + S \times_4 \phi^T) \times_3 s^T\right\|^2$. The optimization is conducted by alternatingly keeping one value of (ϕ, s) fixed and the other variable to minimize $\|J\|^2$. This process is conducted as described in Section 2.4. In the experiments, we set $\varepsilon = 10^{-4}$ to ensure a fast convergence of optimization and remain a good estimation error rate.

$$\sqrt{\sum_{i=1}^{9} \left(s_i^{current} - s_i^{previous} \right)^2} \rightarrow \varepsilon. \tag{6}$$

It should be noted that an estimation achieved by optimizing the difference between the image and its presentation model described above is not only used to find the personal identity vector, but also to handle the illumination of the image. The lighting conditions of images captured in the real world are variant and arbitrary. To handle this situation, the proposed method considers a frontal face image as a model with two variables: a personal identity vector that represents each person's face and the lighting conditions. An iteration is utilized for the optimization of the difference between the real image and the model to calculate the

convergent value of the lighting condition. This convergent value represents the lighting conditions of the image. Therefore, the usage of the ALS algorithm with the Lambertian model allows the application to process the arbitrary lighting conditions of the face image.

In the modeling step, a linear mapping from the identity vector to depth information has already been calculated. Therefore, in the mobile device application, after estimating the identity vector ϕ of the face image, the depth map is constructed using the following equation:

$$d = M\phi + D. \tag{7}$$

Here, d is the depth map that must be estimated, M is the linear mapping parameter matrix calculated in the modeling step, and D is the average value of the depth images of the modeling faces. D must be added to the formula of the reconstruction depth information because the identity vector set, $x = x^T \widehat{w}_x$, and the depth set, $y = y^T \widehat{w}_y$, in the modeling step are all subtracted by their mean values when applying the CCA algorithm.

4.4 Optimization

This part presents the methods used to improve the runtime execution of the code in this program. These optimization is more and more important in implementation on smartphones due to their limitation of processability and computablity.

4.4.1 Analytical simplification for dimension reduction of the face image

This section explains the optimization of dimension reducing operations in 2D space. As mentioned in Section 2.5, to accelerate the process, the face image in 2D space is dimension-reduced by a projection to the matrix space

Fig. 8 performs the directions of two resultant matrices from SVD factorization on the face image. The dimension reduction of the face image is conducted in each mode of the tensor-based function by projecting the face image into the matrix space formed from these two matrices.

Theoretically, the projection of the face matrix to the matrix space should be a complicated compution transposing the matrix I in accordance with the N-mode product. For example, the 2-mode product of a tensor $\mathbf{F} \in \mathbf{R}^{n_1 \times n_2 \times \cdots \times n_p}$ with $U_x^T \in \mathbf{R}^{n_2 \times n_2}$ is conducted as follows:

$$\mathbf{F} \times_2 U_x^T \to U_x^{n_2 \times n_2} . \mathbf{F}^{\,n_2 \times n_1 n_3 \cdots n_p} \to \left(U_x \mathbf{F}\right)^{n_2 \times n_1 n_3 \cdots n_p} \to \left(U_x \mathbf{F}\right)^{n_1 \times n_2 \times n_3 \cdots \times n_p}. \tag{8}$$

This process requires a significant computation of transposing the tensor F and the matrix U_x^T, flattening and reshaping tensor F. However, it should be noted that the face matrix I is analyzed in 2D space, and therefore, transposition only occurs between two dimensions.

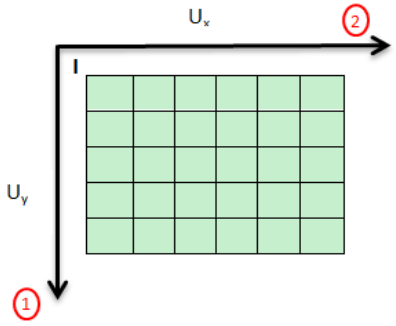


Fig. 8. Reducing dimension of the face image

Consequently, based on the fact that U_y, U_x are the corresponding results of the SVD factorization of the spherical harmonic coefficients on the y and x axis, we can induce the result of the projection.

$$\mathsf{L} = I_{norm} \times_1 \mathsf{U}_y^T \times_2 \mathsf{U}_x^T = U_y^T I_{norm} U_x. \quad (9)$$

This analytical simplification accelerates the implementation of the application significantly because transposition between the rows and columns of large matrices is computationally costly. Furthermore, this formulation also provides the appearance of being 'friendlier' than the tensor form.

4.4.2 Programmable data accessing optimization

It is well-known in programming techniques that processing with a single array of aggregate objects is not as quick as processing two or more arrays having the same length in parallel. For example, the summation $s.c = s.a + s.b$ usually cannot be obtained as quickly as the summation $c = a + b$. In our proposal, s is an aggregate object where the $c, a,$ and b component arrays are the same size as the $c, a,$ and b arrays, respectively. Therefore, we convert the aggregate objects to the predefined-size arrays and accessing its values manually to process in the N-mode SVD calculation. This method improves the processability of not only the basic operations of tensors and matrices but also the transposition of tensors that usually becomes more complicated and costly with the higher-order tensors. The disadvantage of using manual accessing pointers is that it can be more critical in that we must pass the correct number of elements and gaps between the data. Consider a case in which a 2D matrix is stored in memory and there are gaps between the previous row and the next row. That is the data block of the previous row is not adjacent to the one of the next row. If we determine the references of the rows based on the adjacent assumption, the result of the operations will definitely be incorrect. Hence, a range-checking step must be provided to process these cases accurately. The references of each block of data should be determined before any calculation. Nevertheless, with large memory matrices and tensor, this step requires a significant real-time computation. To avoid these punitive situations, matrices and tensors must be stored continuously. This, however, requires a large hardware memory and becomes troublesome for the fast implemention of algorithms on mobile devices. To balance between these two factors, we choose some necessary tensors, not all, to apply the method of manual accessing data. They are the high-order tensors, R (three-dimensional) and S (four-dimensional), that usually require excessive time to calculate in the N-mode product.

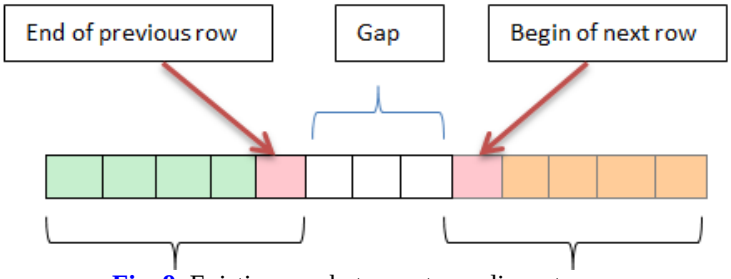


Fig. 9. Existing gap between two adjacent rows

Furthermore, after being estimated in the modeling step, these tensors are loaded in the testing step. Hence, they are easier to be set as contiunous data. We, therefore, forcibly preload these tensors as continuous data in both the computer and smartphone programs.

4.4.3 Optimization Result

The experimental result shows that this technique significantly enhances the computability of the program. **Fig. 10** performs a comparison of the computation of the ALS algorithm using the suggested optimization with one using aggregate objects in terms of time requirement. In this experiment, we implement the 3D reconstruction with five faces and iterate 20 times to obtain the average runtime of the ALS realization in one loop iteration. The runtimes presented in **Fig. 10** in milliseconds are the average values of the processing runtimes for five faces. The detailed result of this experiment is summarized in **Table 1**. The runtime is improved by more than a factor of ten. Moreover, the application is remain relatively safe in terms of memory reliability.

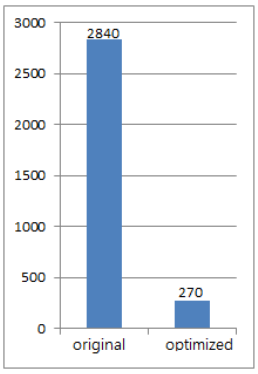


Fig. 10. Computability of the optimized and original versions.

Table 1. Comparison cost in the original and optimized program.

Program	Face 1	Face 2	Face 3	Face 4	Face 5
Original (ms)	2862	2840	2826	2840	2835
Optimized (ms)	276	260	258	297	258

5. Performance and time evaluation of 3D facial reconstruction

In this section, we describe the results of a reconstruction application with photographs taken using the built-in camera and images loaded from the internal memory on the Android device. An evaluation of the required processing time is provided to demonstrate the processability of the application.

5.1 Performance

The application detects the face area of an image and crops it for use as a single frontal face input. A high-resolution 3D face is reconstructed from that frontal face under any lighting conditions. **Fig. 11** shows the application interface.

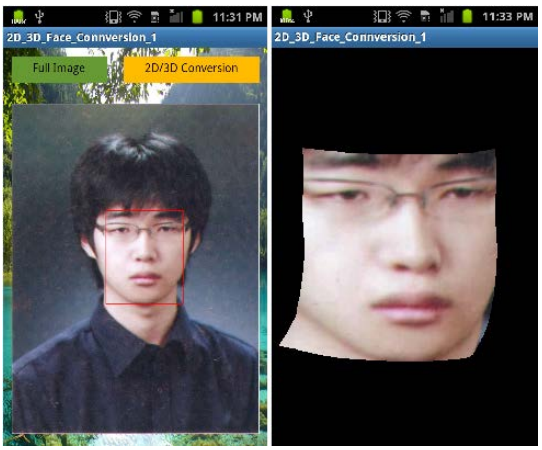


Fig. 11. The designed user-interface of the application

To illustrate the implementation quality with both good and arbitrary illumination conditions, we determined the reconstruction results of both an image loaded from internal memory with good lighting and resolution, and an image captured from a built-in camera with arbitrary lighting conditions and limited resolution. **Fig. 12** and **Fig. 13** show the results of the implementation of the image in **Fig. 7** that was loaded from the internal memory of the mobile device during the experiment.

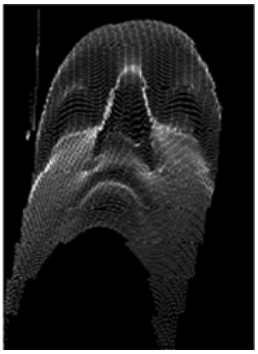


Fig. 12. Depth map in point map

Fig. 13. Visualization of 3D face

Fig. 12 shows a depth map as 3D points in a 3D space. These points represent the x, y, and z coordinates of each pixel converted into a 3D space by applying mapping. Upon the addition of the texture information from the input image, we obtained the final 3D image of a single frontal face image. This is the result of our proposed implementation. A visualization of the 3D image is presented in **Fig. 13**. **Fig. 15** shows a 3D reconstructed map under different viewing angles of a face detected from the image in **Fig. 14**. This was captured from the built-in camera.

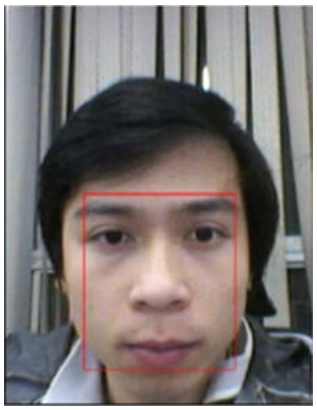


Fig. 14. Captured image

Fig. 15. 3D face from different viewpoints

Table 2 shows the effect of the optimized application. It executes significantly faster than the original method. It is also faster than the development code in the Matlab environment. The result and the 3D facial reconstruction error for a test image from the database proves that the optimized program developed in a C/C++ environment is 38.2% faster and maintains the accuracy of the Matlab program.

Table 2. Comparison of the execution times.

Environment	C/C++		Matlab
	Original	Optimized	
Image size	480x640 pixels	480x640 pixels	480x640 pixels
Time cost	2820ms	204ms	330ms

5.2 Test time evaluation

The evaluation conducted herein utilized the recent Samsung Galaxy S2 smartphone and the code was used without optimization. However, to improve the computation ability of the application, we used native C/C++ code instead of a Java development environment. Java code must run on a Java Virtual Machine, whereas native C/C++ code does not require the use of such a virtual machine. Android runs on a Dalvik Virtual Machine. In this work, we used many tensors and matrices with complex operators, memory accesses, and heap memory allocations, and thus, as recommended by Lee et al. [22], it is better to use native C/C++ code than to run Java code in a Dalvik Virtual Machine.

To evaluate the execution time of this application on an Android smartphone, we tested a 3D facial reconstruction from a camera by capturing a face five times and obtained the average value of the processing time. The experimental results show that the application runs quickly and robustly, and manages the computational problems well. Specifically, it takes approximately 270 microseconds for the reconstruction step and 1.13 seconds for all parts of the application on an Android smartphone. **Table 3** provides a comparison of the experimental devices used. **Table 4** shows the results of implementing the reconstruction on a desktop computer using a C/C++ environment and on a Samsung Galaxy S2 smartphone. The computer utilized was a 3.00 GHz AMD Phenom II X6 1075T processor and the Galaxy S2 had a Dual-core 1.2 GHz Cortex-A9. The computer was much more powerful than the

smartphone, as shown in **Table 3**, and consequently it required more time to execute the proposed method on a smartphone, as shown in **Table 4**.

Table 3. Setup of the experimental devices.

Device	PC	Smartphone
Chip	X6 3Ghz	Duo-core 1.2Ghz
RAM	4GB	1GB
Development Environment	Visual studio with C/C++	Android with native code support

Table 4. Reconstruction cost in C/C++ and in an Android environment.

Source	Test image		Captured image	
Size	480x640 pixels		480x640 pixels	
Environment	C/C++	Android	C/C++	Android
Time cost	204ms	312ms	212ms	316ms

6. Conclusion

In this paper, the rapid implementation of a CCA-based 3D facial reconstruction from a single frontal face image under arbitrary illumination conditions on an Android mobile device has been proposed. With the proposed reconstruction system, we can obtain a high-resolution 3D face corresponding to the frontal face of the input image. The optimization based on analytical simplification and a heuristic programming method significantly improves the computational performance of the implementation compared to the previous effort. Moreover, separating the model training and 3D reconstruction allows the application to be faster, lightweight, portable, and suitable for an application on a mobile device. Further development of this application will include improving the performance of the application and inserting an engine for the processing of the head, ears, and other facial features. These improvements remain as future work.

References

[1] Y. Lee, D. Terzopoulos, K. Waters, “Realistic modeling for facial animation,” *SIGGRAPH*, pp. 55-62, 1995. Article (CrossRefLink).

[2] W. Lee, P. Kalra, M. N. Thalmann (1997), “Model Based Face Reconstruction for Animation,” *MMM*, pp. 323-338, 1997. Article (CrossRefLink).

[3] G. Sannier, M. N. Thalmann, “A flexible texture fitting model for virtual clones,” In *Proc. of Computer Graphics International, IEEE Computer Society*, pp. 66-99, 1997. Article (CrossRefLink).

[4] R. Zhang, P. Tai, J. E. Cryer, M. Sha, “Shape from shading: a survey,” *PAMI, 21(8)*, pp. 690-706, 1999. Article (CrossRefLink).

[5] Y. Hu, D. Jiang, S. Yan, L. Zhang, H. Zhang, “Automatic 3D Reconstruction for Face Recognition,” In *Proc. of 6th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pp. 843-848, 2004. Article (CrossRefLink).

[6] Y. Hu, Y. Zheng, Z. Wang, "Reconstruction of 3D face from a single 2D image for face recognition," In *Proc. of 2nd Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 217-222, 2005. Article (CrossRefLink).
[7] K. N. Choi and M. C. Mozer, "Recovering facial pose with the EM algorithm," pp. 2073-2093, 2002. Article (CrossRefLink).
[8] Y. Zhou, L. Gu, and H. Zhang, "Bayesian tangent shape model: Estimating shape and pose parameters via bayesian inference," In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, vol.1, pp. 109-116, 2003. Article (CrossRefLink).
[9] S.W. Park, J. Heo, M. Savvides, "3D face reconstruction from a single 2D face image," In *Proc. of IEEE Computer Society Conference*, pp. 1-8, 2008. Article (CrossRefLink).
[10] C. Xin, T. Faltemier, P. Flynn, K. Bowyer, "Human Face Modeling and Recognition Through Multi-View High Resolution Stereopsis," In *Proc. of Conf. on Computer Vision and Pattern Recognition Workshop*, pp. 50, 17-22 June, 2006. Article (CrossRefLink).
[11] M.S. Hossain, M. Akbar, J.D. Starkey, "Inexpensive construction of a 3D face model from stereo images," In *Proc. of 10th Int. Conf. on Computer and information technology*, pp.1-6, 2007. Article (CrossRefLink).
[12] M. Lee, C.-H. Choi, "Fast facial shape recovery from a single image with general, unknown lighting by using tensor representation," *Pattern Recognition*, vol.44 no.7, pp. 1487-1496, 2011. Article (CrossRefLink)
[13] M. Reiter, R. Donner, G. Langs, H. Bischof, "3D and infrared face reconstruction from RGB data using canonical correlation analysis," In *Proc. of Int. Conf. on Pattern Recognition*, vol.1, pp. 425-428, 2006. Article (CrossRefLink)
[14] W.B. Lee, M.H. Lee, I.K. Park, "Photorealistic 3D face modeling on a smartphone," In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, vol., no., pp.163-168, 20-25 June, 2011. Article (CrossRefLink).
[15] C. Wang, M. Bao, T. Shen, "3D model reconstruction algorithm and implementation based on the mobile device," *J. Theoretical and Applied Information Technology*, vol.46, no.1, pp. 255-262, 2012. Article (CrossRefLink).
[16] P.J. Phillips, P.J. Flynn, T. Scruggs, K.W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, W. Worek, "Overview of the face recognition grand challenge," In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 947- 954, 20-25 June, 2005. Article (CrossRefLink).
[17] Z. Lei, Q. Bai, R. He, S.Z. Li, "Face shape recovery from a single image using CCA mapping between tensor space," In *Proc. of IEEE Conf. CVPR*, pp.1-7, 23-28 June, 2008. Article (CrossRefLink).
[18] R. Basri, D.W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Trans. Pattern Analysis and Machine Intelligence 25*, pp. 218-233, 2003. Article (CrossRefLink)
[19] D. Frolova, D. Simakov, D. Basri, "Accuracy of spherical harmonic approximations for images of Lambertian objects under far and near lighting," *European Conf. on Computer Vision*, 2006. Article (CrossRefLink).
[20] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," In *Proc. of IEEE Conf. CVPR*, vol.1, pp. 511-518, 2001. Article (CrossRefLink).
[21] G. Bradski, A. Kaehler, "Learning OpenCV: Computer Vision in C++ with the OpenCV Library," *O'Reilly Media, Inc.*, 2013. Article (CrossRefLink).
[22] S. Lee, J.W. Jeon, "Evaluating performance of Android platform using native C for embedded system," In *Proc. of Int. Conf. on Control, Automation and Systems*, pp 1160-1163, 2010. Article (CrossRefLink).
[23] A. Cichocki, R. Zdunek, A. H. Phan, S. I. Amari, "Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation," *Wiley*, 2009. Article (CrossRefLink).

Phuc Huu Truong received the B.E. degree in automation from the HCM University of Technology, Vietnam, in 2011, and the M.S. degree from Electrical Engineering from Kookmin University, Seoul, South Korea, in 2013. In 2013, he joined the Korea Institute of Industrial Technology (KITECH) as a researcher. He is currently pursuing a Ph.D. degree in Electrical Engineering at Kookmin University. His research interests include computer vision, pattern recognition, machine learning, and mobile platforms.

Chang-Woo Park received B.S and M.S. from the Electronic Engineering of Kookmin University in 2012 and 2014, respectively. He is currently a research assistant at Kookmin University. His research interests include pattern recognition, mobile robot, and mobile platforms.

Minsik Lee received the B.S. and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Korea, in 2006 and 2012, respectively. He is currently a BK21 Assistant Professor in the Graduate School of Convergence Science and Technology, Seoul National University, Suwon, Korea. His research interests include computer vision, pattern analysis, machine learning, image processing, and their applications.

Sang-Il Choi received the B.S. degree in the division of electronic engineering from Sogang University, Seoul, Korea, in 2005 and the Ph.D. degree from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, in 2010. He was a Postdoctoral Researcher in the BK21 Information Technology, Seoul National University, in 2010 and in the Institute for Robotics and Intelligent Systems of Computer Science Department, University of Southern California, Los Angeles, until August of 2011. He is currently an Assistant Professor with the Department of Applied Computer Engineering, Dankook University, Gyeonggi-do, Korea. His research interests include pattern recognition, feature extraction and selection, machine learning, computer vision, and their applications.

Sang-Hoon Ji received his B.S. and M.S. degrees in Control and Instrumentation Engineering and his Ph.D. degree in Electrical Engineering and Computer Sciences from Seoul National University, Seoul, Korea in 1995, 1997, and 2007, respectively. From 1997 to 2002, he was a Research Engineer at IAE, Yongin, Korea and from August 2007 to September 2008 he worked as Deputy General Manager at Doosan Infracore Ltd., Yongin, Korea. Since October 2008, he has worked with Robot R&D Group at Korea Institute of Industrial Technology, Ansan, Korea, where he is currently a Principal Researcher. His research interests include multi-agent robot systems, sensor based robotics, robot S/W platform, and medical robots.

Gu-Min Jeong received the B.S. and M.S. degrees from the Dept. of Control and Instrumentation Eng., Seoul National University, Seoul, Korea, in 1995 and 1997, respectively, and Ph.D. degree from School of Electrical Eng. and Computer Science, Seoul National University, Seoul, Korea in 2001. He was a Senior Engineer at NeoMtel, Korea from 2001 to 2004 and a Manager at SK Telecom, Korea from 2004-2005. Also, from 2011 to 2013, he was a Visiting Associate Professor with the Department of Computer Science, University of California Irvine, Irvine. Currently, he is an Associate Professor of School of Electrical Engineering, Kookmin University, Seoul, Korea. His research area includes applied embedded systems, pattern recognition, and control systems.