# Face Recognition Technology for Ubiquitous Computing Environment

Kanghun Jeong, Seongrok Hong, Ilyang Joo, Jaehoon Lee, and Hyeonjoon Moon

School of Computer Engineering, Sejong University, Seoul, Korea
hmoon@sejong.ac.kr

**Abstract.** In this paper, we explore face detection and face recognition algorithms for ubiquitous computing environment. We develop algorithms for application programming interface (API) suitable for embedded system. The basic requirements include appropriate data format and collection of feature data to achieve efficiency of algorithm. Our experiment presents a face detection and face recognition algorithm for handheld devices. The essential part for proposed system includes; integer representation from floating point calculation, optimization of memory management scheme and efficient face detection performance on complex background scene.

**Keywords:** ubiquitous computing environment, face recognition, face detection, application programming interface, algorithm optimization.

## 1 Introduction

In recent years, face detection and recognition technology has been developed rapidly. The need for biometric security system had increased for ubiquitous computing environment. Face recognition technology maintains high security level while providing convenience for both human and computer. Therefore, it is necessary to optimize the face recognition system by maintaining recognition accuracy while decreasing computational complexity. However, most of the hand-held devices are hard to satisfy such factors because of limited configuration of memory and CPU power.

The major factors of the face recognition system are feature size and the processing efficiency. Generally, processing time is governed by geometric progression which is the dimensionality of feature data in the case of mobile-based devices. It is essential to optimize data structure while maintaining a reasonable recognition performance.

In this paper, we explore various algorithms for face detection and recognition algorithms. Experiments include normalization of a face database to increase recognition performance through various pre-processing methods. We propose a novel algorithm for automatic face detection and post-processing method to further improve the face recognition performance.

C. Stephanidis (Ed.): Universal Access in HCI, Part II, HCII 2009, LNCS 5615, pp. 365–373, 2009.
© Springer-Verlag Berlin Heidelberg 2009

## 2 Sejong Face Database (SJDB)

Generally, face recognition system requires face image data that used for learning and features are normalized and registered for face recognition process. These face image data are collected for some time intervals in various conditions [1]. We have collected Sejong face database for this experiment and images are called 'session' images. There are several session images in the Sejong face database. A set of images of each person are used for learning and other sets are used for face recognition system. A session contains three images for each person (Figure 1).

**Fig. 1.** Sejong face image database (SJDB)

The session images of each person are collected one or two images a day in the same physical setup such as pose and lightning. Session images of N different people are collected without distinction of sex. In this way, we collected 100 different 'session' about 70 male and 30 female. Each session contains three frontal images which were collected with similar illumination condition (170~220 lux) [2][3].

## 3 Face Detection and Face Recognition

For any face recognition system noise-data is a primary factor of degradation. Therefore, the process for remove noise is essential to improve recognition accuracy [3][4]. This process is called pre-processing and it begins with geometric transformation (rotation, translation, and rescaling). The correction of the geometric transformation uses the intermediate point of the eyes and mouth. We remove the unnecessary data of the outer face region using an oval mask. These processes can be more accuracy controlled as shown in Figure 2.

The pre-processed image is properly modified with several feature point (eyes, nose, mouth) to have equal dimension regardless of personal face size [5] (Figure 3). The face images for this experiment has horizontal versus vertical ratio of 3:4. As a result, face images are modified to 1200*1600, 120*160, and 30*40 size.

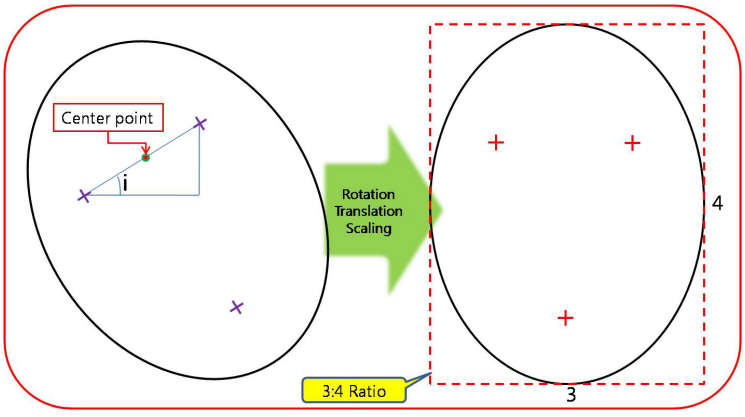



**Fig. 2.** Pre-processing with the mask and three points information

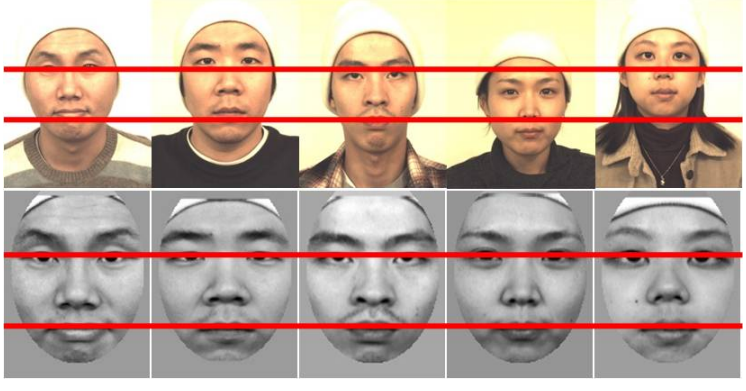


**Fig. 3.** Sejong face database (SJDB) and pre-processed face data

The pre-processing time is shown in Table 1 which was performed based on Intel Pentium 4 computer with 3.0GHz and 1GByte configuration. In this paper the pre-processing was performed with fully automatic setup with Adaboost algorithm [6][7].

**Table 1.** Pre-process time for one face data

| Image size / Processing time (ms) | 1200 * 1600 | 120 * 160 | 30 * 40 |
|---|---|---|---|
| Processing time (ms) | 1639.75 | 449.02 | 414.24 |

Understanding it in Table 1., if image size reduced ratios of the decrease of time are different. Time when it is necessary to preprocess size of 120 * 160 and an image of 30 * 40 has few differences, the number of image pixel used for face recognition has many differences. Data used for collecting recognition with much number of pixels increase, and recognition performance improves. The image with size of 120 * 160 does the best performance reference of Table 1.

A face image is converted into a pre-processed image for face recognition with normalized pixel value as well as geometric transformation. In this experiment, we have applied series of algorithms including histogram equalization and filtering. As for the image, noise data was removed and the feature data is emphasized. In addition, intensity information of the image was redistributed through histogram equalization. The structure of our face recognition system is shown in Figure 4.

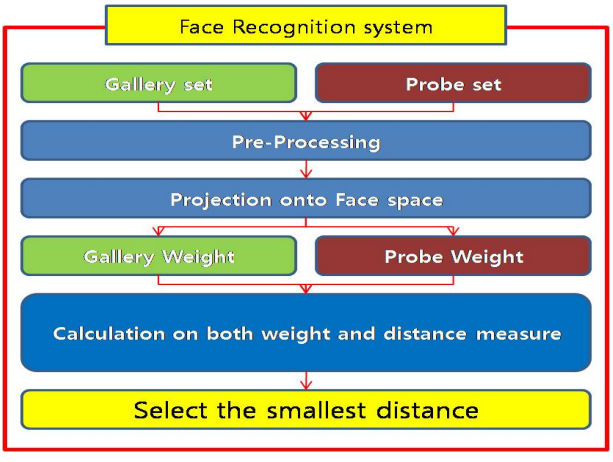



**Fig. 4.** Structure of proposed face recognition system

We have used principal component analysys (PCA) [8] and linear discriminant analysis (LDA) for feature extraction [9]. As for the feature data, the change of recognition rate is presented with different cutoff ratio (percentage of feature vectors used) as shown in Table. 2. We set the cutoff ratio at 80% and designed our face recognition system which we considered as most appropriate performance. We have measured the distance between feature vectors based on *L2 norm* (Euclidian distance).

**Table 2.** A face recognition result by the change of image size and cutoff ratio

| Cutoff \ Image size | 1200 * 1600 | 120 * 160 | 30 * 40 |
|---|---|---|---|
| 100 % | 96 % | 98 % | 97 % |
| 80 % | 97 % | 98 % | 96 % |
| 70 % | 94 % | 95 % | 91 % |
| 50 % | 91 % | 87 % | 82 % |

The choice of the automatic face detector was based on Adaboost algorithm which was trained using FERET [10], XM2VTS [11], CMU PIE [12] and Sejong face database.

The size of learning data is using face (30x30), eyes (10x6) and mouth (20x 9) pixels (Figure 5). The 3,513 positive data and 10,777 negative pieces of image data are collected and used (Figure 6). As a result, we succeeded in the face detection from near frontal face images ranging from 0 to 15 degrees. Figure 7 is the structure of the Adaboost classifier used in this experiment [7].

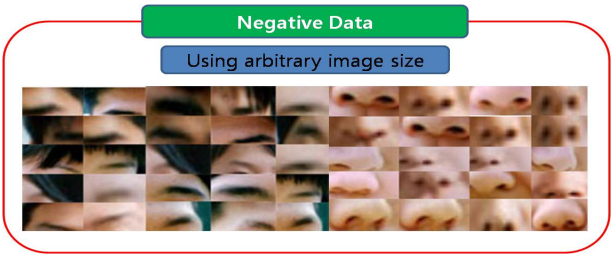



**Fig. 5.** Negative training data

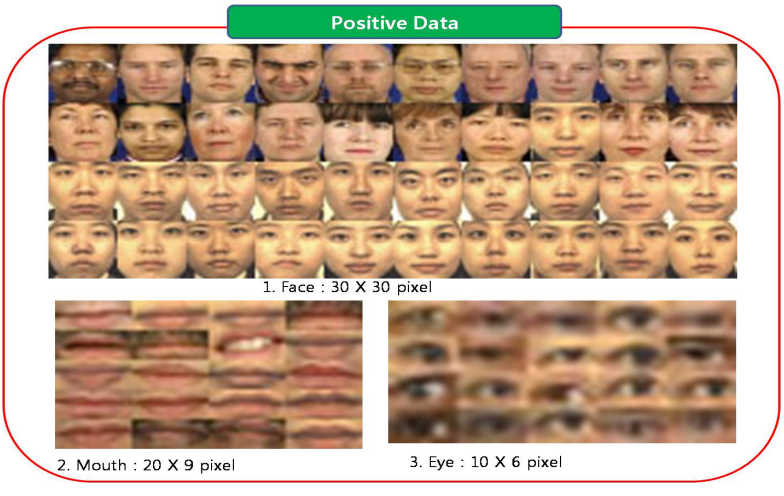



**Fig. 6.** Positive training data

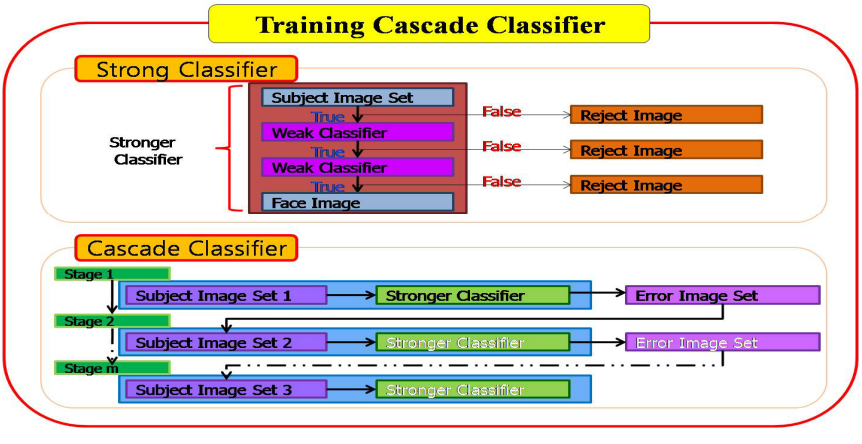



**Fig. 7.** Adaboost classifier

For learning the face detection data using Adaboost algorithm, negative (wrong) data is equally important as positive (correct) data. After appropriate training, the face detector can locate feature points (the center of each eyes and the center of the mouth) from a various rotated angle and front face image. After pre-processing of face image, face recognition takes about 7 fps (frame per second) which is reasonable for real-time processing. The total processing time requires 1.1 fps (frame per second) including the face detection. Face detection and face recognition process show rate of 87% detection and 95% recognition percentage each.

## 4 Face Recognition Application Programing Interface (API)

We have implemented four major face recognition functions into application programming interface (API) to performed general face recognition experiment. API is composed of face detection, face recognition, face similarity, and face evaluation module designed based on C/C++ as shown in Table 3.

**Table 3.** API for Face Recognition Function Modules

| Function | Description |
|---|---|
| Face Detection | Face region and feature point detection for input image |
| Face Recognition | After pre-processing, face recognition on detected face image. |
| Face Similarity | Face similarity calculation between probe image and gallery image. |
| Face Evaluation | Evaluation of face detection and face recognition algorithms |

Generally, face recognition algorithms are expressed by normalized numerical value. Such real number based algorithms are used for graphical algorithms such as image processing, 3D rendering, etc. There exists noticeable performance drops between floating and integer number based calculation. Moreover, this difference appears greatly on embedded system without floating point unit (FPU).

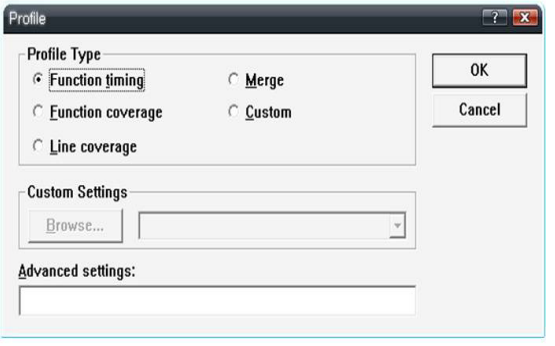
Since face recognition API use floating point based calculation algorithm, performance degradation can be significant. In order to reduce these differences, integer number type was used with fixed point method which is a numerical analysis method to store fraction's part of the decimal and the integer number separately.

Fixed-point method was used for API to produce faster than existing program with floating algorithm. We designed fixed decimal point algorithm is designed to perform faster than floating point algorithm. There is proved measurement of a runtime through simple C program in Figure 8.

Integeral fragmentary performance can be verified through profiling. In practice, performance can be evaluated through integeral conversion which uses fixed point algorithm. Face image data was used for performance evaluation with 120x160 pixels. Evaluation unit compare each module during the process. Change of total arithmetic velocity depends on integeral transform considerably.

```
Program Statistics
------------------
    Command line at 2008 Nov 04 09:45: "E:₩0. 바탕화면₩0. Coding₩5. 정수화 관련₩[081026_
    Total time: 37.984 millisecond
    Time outside of functions: 0.286 millisecond
    Call depth: 15
    Total functions: 440
    Total hits: 906
    Function coverage: 50.0%
    Overhead Calculated 499
    Overhead Average 499

Module Statistics for test2.exe
-------------------------------
    Time in module: 37.698 millisecond
    Percent of time in module: 100.0%
    Functions in module: 440
    Hits in module: 906
    Module function coverage: 50.0%
```




```
        Func          Func+Child          Hit
        Time   %         Time      %      Count  Function
-----------------------------------------------------------
      12.318  32.7       12.318   32.7        1 NMul(double,double) (main.obj)
      11.154  29.6       11.154   29.6        1 NMul(float,float) (main.obj)
       2.556   6.8        2.556    6.8        1 FMul(double,double) (main.obj)
       2.520   6.7        2.520    6.7        1 FMul(float,float) (main.obj)
       2.057   5.5        2.057    5.5        1 FDiv(float,float) (main.obj)
       2.056   5.5        2.056    5.5        1 FDiv(double,double) (main.obj)
       0.966   2.6        0.966    2.6        1 NDiv(double,double) (main.obj)
       0.841   2.2        0.841    2.2        1 NAdd(double,double) (main.obj)
       0.841   2.2        0.841    2.2        1 NSub(double,double) (main.obj)
       0.482   1.3        0.482    1.3        1 NDiv(float,float) (main.obj)
       0.360   1.0        0.360    1.0        1 NAdd(float,float) (main.obj)
       0.360   1.0        0.360    1.0        1 NSub(float,float) (main.obj)
       0.295   0.8        0.295    0.8        1 FSub(double,double) (main.obj)
       0.280   0.7        0.280    0.7        1 FAdd(double,double) (main.obj)
       0.280   0.7        0.280    0.7        1 FAdd(float,float) (main.obj)
       0.280   0.7        0.280    0.7        1 FSub(float,float) (main.obj)
       0.004   0.0        0.004    0.0        2 operator delete(void *) (delop.obj)
       0.003   0.0        0.011    0.0       26 std::locale:: Init(void) (locale0.obj)
```

**Fig. 8.** Function profiling result (performance increase by the fixed-point use)

**Table 4.** 7,000,000 times of profiling repetition results

| Intel ® Core™2 Quad Q9300 | | | |
|---|---|---|---|
| | Integer number | Floating number | Double number |
| Addition | 62.1 | 89.1 | 255.1 |
| Subtraction | 78.2 | 87.6 | 277.6 |
| Multiplication | 113.3 | 447.2 | 997.1 |
| Division | 311.2 | 490.5 | 817.7 |

**Table 5.** 1,000,000 times of profiling repetition results

| Intel ® Pentium – 233 | | | |
|---|---|---|---|
| | Integer number | Floating number | Double number |
| Addition | 39.246 | 38.661 | 40.044 |
| Subtraction | 38.242 | 38.506 | 40.400 |
| Multiplication | 39.156 | 528.966 | 528.861 |
| Division | 197.004 | 579.077 | 574.697 |

This fact is that because additional arithmetic is contained through integeral transform, and performance improvement is gauranteed if it can be applied on embedded system as shown in Table 4 and Table 5.

## 5 Conclusion

The face recognition system is essential in multiple biometric system because it provides the only non-contact biometric features and user friendly information which can be processed by both human and computer. This is very important feature in case of communication problem caused by network failure which prohibits database access for border control applications. Recently, face detection and recognition technology can be build into a embedded system for closed-circuit television (CCTV) related applications. But this system should respond to numerous conditions including various pose and lighting which is challenging for face recognition system. Generally, a face recognition system contains two major function which is face detection and face recognition. We produce several face detection and recognition algorithms into application programming interface (API) for various biometric applications. We present performance evaluation for integral transformed calculation which is optimize for embedded system suitable for ubiquitous computing environment. We have trained face detection and face recognition modules based on facial recognition API using Sejong face database. Proposed face detection and face recognition modules are integral transformed which can be calculated faster, and optimized for embedded system. Our experimental result shows that we can minimize the run-time by decreasing computational complexity while maintaining reasonable accuracy which is applicable for ubiquitous computing environment.

**Acknowledgement.** This work was supported by the Seoul R&BD Program (10581).

## References

1. Papatheodorou, T., Rueckert, D.: Evaluation of 3D Face Recognition Using Registration and PCA. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 997–1009. Springer, Heidelberg (2005)
2. Tusk, M., Pentland, A.: Eigenfaces for recognition. J. Cognitive Neuroscience 3, 71–86 (1991)

3. Moon, H., Phillips, P.: Computational and Performance Aspects of PCA-Based Face-Recognition Algorithms. Perception 30, 303–321 (2001)
4. Sim, T., Baker, S., Bsat, M.: The CMU Pose, Illumination, and Expression Database. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(12), 1615–1618 (2003)
5. Phillips, P., Moon, H., Rauss, P.: The FERET Evaluation Methodology for Face Recognition Algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(10), 1090–1104 (2000)
6. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, pp. 511–518 (2001)
7. Viola, P., Jones, M.: Robust Real-time Object Detection. In: Second International Workshop on Statistical and Computational Theories of Vision, July 13 (2001)
8. Turk, M., Pentland, A.: Eigenfaces for recognition. J. Cognitive Neuroscience 3, 71–86 (1991)
9. Yambor, W.S.: Analysis of PCA-based and Fisher Discriminant-based Image Recognition Algorithms. Technical report, CSU (June 2000)
10. FERET Database. NIST (2001), http://www.itl.nist.gov/iad/humanid/feret/
11. XM2VTS Database, http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/
12. Sim, T., Baker, S., Bsat, M.: The CMU Pose, Illumination, and Expression Database. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(12), 1615–1618 (2003)