Enabling EBGM Face Authentication on mobile devices

Y.S. Moon, K. H. Pun, K.C.Chan, M.F. Wong, T.H.Yu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, HONG KONG
[ysmoon, khpun, kcchan,mfwong3, thyu4]@cse.cuhk.edu.hk

Abstract

This paper presents a systematic optimization strategy to implement the complicated EBGM face recognition on low processing power mobile devices We propose a tailor-made fixed point arithmetic and various memory access optimization techniques to speed up an EBGM authentication process from ~550s down to ~1s in a typical Intel PXA255 400 MHz mobile platform. The result shows that real time face recognition can be completed without any noticeable accuracy loss. This finding not only provides a guideline for porting various biometric applications in the mobile systems, but also exploits new opportunities for different mobile e-commerce applications with biometric identity checking.

1. Introduction

2D Face recognition systems are often classified by their representation scheme. The two major classes of systems are global feature based and local feature based systems. In global feature based systems, face images are treated as a whole and statistical information from the entire face is extracted. Most systems in this class involve finding an easily separable subspace. By projecting the high dimensional face image to a well-selected subspace of lower dimension, an efficient and possibly discriminating representation is acquired. Various subspace selection methods, including Evolution Pursuit (EP) [1], Independent Component Analysis (ICA) [2], Principal Component Analysis (PCA) [3] and Linear Discriminant Analysis (LDA) [4] are commonly used, with the latter two being the most popular ones due to their simplicity and reasonable performance.

In local feature based system, the relative size of and distance between various facial components like eyes, nose and mouth are measured and used as features. Local features are often extracted by means of filter convolution, such as Gabor filters. The most representative methods in this class include Elastic Bunch Graph Matching (EBGM) [5] and Local Feature Analysis (LFA) [6].

In recent years, we see a proliferation of mobile devices such as PDAs and cell phones as well as applications on them. Individuals use them for electronic transactions such as phone banking and stock trading, while enterprises issue them as a means of access to the corporate network. Security concerns with mobile devices have become an imminent issue as a poorly protected mobile device may expose sensitive data such as passwords and credit card information, or may become a security hole of the entire corporate network. To provide authentication and authorization checking in accessing the mobile devices, biometrics, especially face authentication, seems to be a good candidate. However, heavy computation requirement in authentic face authentication algorithms hinders the use of secure face recognition in the mobile world.

Due to the portability purpose, inherent limitations such as size, cost and power consumption still persist for all mobile devices. To conserve battery power and chip size, typical mobile processors such as the Intel XScale, have a comparatively small cache size (Figure 1) and do not come with a Floating Point Unit (FPU). A small cache size means data and instructions are swapped out of cache more frequently and much slower off-chip memory will be accessed. To tackle such problems, optimization must focus on speeding up the computational operation and improving the cache performance.

In our previous work, we have already implemented a real-time mobile face recognition system using PCA technique [13]. However, its intrinsic slow training process (registration process) prohibits its use in

changing environments. In this paper, we describe an alternative, optimized EBGM algorithm, which is more accurate, does not rely on training process and can be applied in both registration and verification processes.

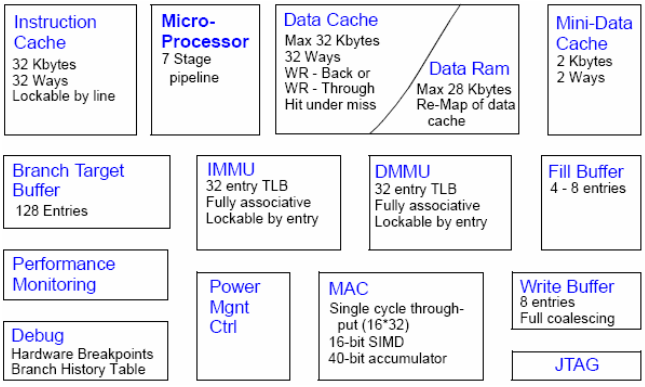


Figure 1. Intel XScale Microarchitecture feature [7]

This paper is organized as follows: Section 2 describes the working principle of EBGM face recognition and outlines the major components of EBGM authentication. Section 3 presents our optimization techniques which are addressed to the weakness of general mobile devices. In section 4, experimental results are illustrated to show our improvements of the face authentication system against the baseline software. Finally, Section 5 gives a brief conclusion.

2. EBGM System Overview

Elastic Bunch Graph Matching [5] represents faces using a set of local features located on an elastic (deformable) graph. Responses to a set of Gabor filters are collected for each node on the graph and stored in the face template. Both graph configuration and local Gabor features are taken into account during matching. Due to its local and flexible nature, EBGM are in general less susceptible to variations in lighting, face position and expression.

EBGM can be divided into 4 stages as shown in Figure 2:

1. Image processing
2. Facial Landmark Localization
3. Gabor Feature Extraction
4. Template matching

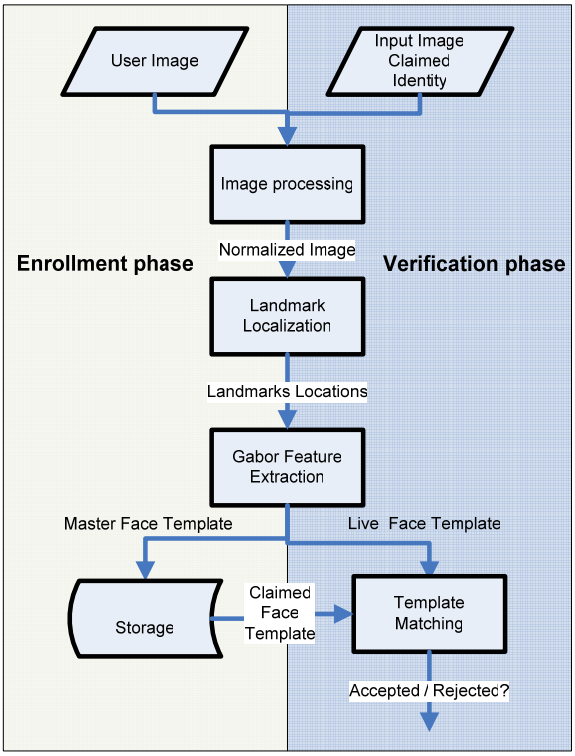


Figure 2. EBGM authentication process

During Image processing, face images are normalized to reduce the variations among them. The normalization routine performs mean centering, edge smoothing, geometric normalization, masking and pixel normalization on the face image [8]. And finally a 128x128 gray scale pre-processed image is generated from a 256 x 384 input image.

After the pre-processing, locations of such facial landmarks as eyes, nose and mouth are found in the landmark localization stage. This stage contains two steps. First, rough estimates of the landmark locations are obtained based on known landmarks, such as the eyes. Then the estimate is refined by Gabor jet comparisons. For the EBGM algorithm, 40 complex Gabor wavelets (or 80 real/imaginary pairs) of different sizes, wavelengths and orientations are used.

During the feature extraction stage, Gabor Jets are extracted from the normalized face image at the landmark locations found in the previous stage. These Gabor features, together with the locations, are stored in a structure called face graph which is stored in the template database for future recognition use and the original face image is then discarded. Figure 3 shows

an example of resulting face graph generated from an input face image.

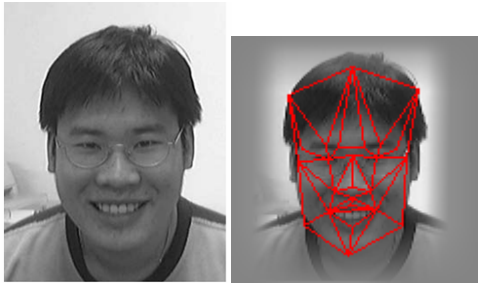


Figure 3. Original image (left) and resulting EBGM Face Graph (Right)

After a face graph has been created for a novel face image, its similarity with another registered face graph (template) can be computed. Given two face graphs, G and G', their similarity L can be computed as:

$$L(G,G') = \frac{1}{M}\sum_{i=0}^{M} S_D(J_i, J_i')$$

where M is the number of landmarks, and J_i and J_i' are jets from the i-th landmarks of graphs G and G'. SD is a similarity measure for Gabor jets defined by:

$$S_D(J,J') = \frac{\sum_{j=0}^{N} a_j a_j' \cos(\Phi_j - (\Phi_j' + \vec{d}\cdot\vec{k}_j))}{\sqrt{\sum_{j=0}^{N} a_j^{\,2} \sum_{j=0}^{N} a_j'^{\,2}}}$$

where N is the number of Gabor filters, and a_j (a_j') and Φ_j (Φ_j') are the magnitudes and phases of the jth filter response from Gabor Jets J(J'). $\vec{d}$ is the estimated displacement between the two jets and is the spatial frequency of the jth filter. The term is used to compensate the phase shifts caused by the displacement, leading to a phase sensitive similarity function. A more detailed discussion on the EBGM algorithm can be found in [9].

3. Optimization Techniques

3.1 Optimization Overview

Optimization of the EBGM algorithm can be classified into two categories

i) Computation Optimization – We address on the weak arithmetic power of mobile processors and re-implement a set of new arithmetic operations to speed up the computational speed. Related techniques include Gabor mask values pre-computation, trigonometric function lookup table and fixed point arithmetic.

ii) Memory Optimization – We address on improving the communication efficiency between the core, on-chip cache and off-chip memory, so that data utilization in cache can be as full as possible. Related techniques include tuning the caching policy, reconstructing the data using suitable data structure (1 D image array / Multiple Gabor mask data structure), array alignment to suit the cache block loading size, etc.

In the following sub-sections, we will focus on some key techniques that we have employed in our mobile system.

3.2 Fixed point arithmetic

Since mobile devices do not have floating point units, computation of floating point operations by using software emulation becomes a burden for mobile devices. To speed up the computation, fixed point arithmetic is applied.

A fixed point data type is implemented using a 32-bit integer (built-in C type "int"). The fixed point representation consists of three parts: sign bit, integer bits and fraction bits. The number of width assigned to the integer part is called Integer Word Length (IWL). Similarly, the number of fraction bits is called Fractional Word Length (FWL). IWL determines the largest possible range that can be represented by a fixed-point number, while FWL determines the precision. [12]

With a limited number of bits to represent the data range, overflow or underflow might occur during the numeric computation. To preserve the required precision, a range estimation process is done to determine the fixed point format for our EBGM implementation.

The range estimation keeps track of the maximum absolute integral value for each floating variable in different stages of EBGM processes and the range estimation result conducted using FERET database [11] are summarized in Figure 4. It is found that 13-bit representation for Integral part is sufficient to represent all the integral values accurately. Hence, we applied a (IWL=13, FWL=18) fixed point representation for our initial stage optimization as illustrated in Figure 5.

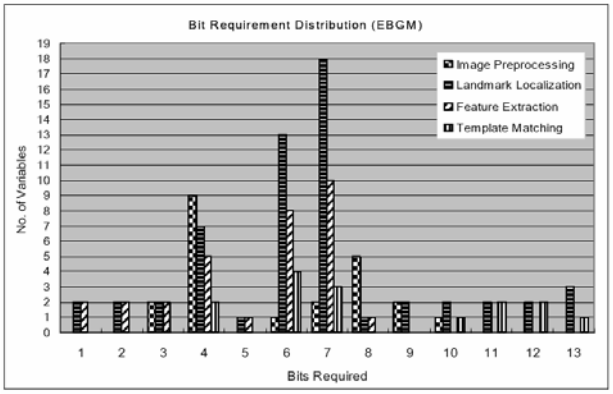


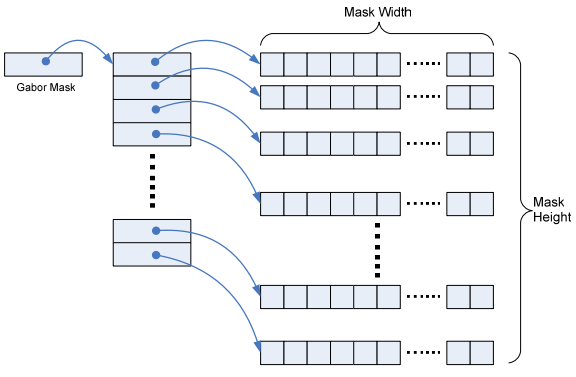
Figure 4. Number of Integral bits required for different floating point variable across the EBGM program.

1 sign bit	IWL=13	FWL=18

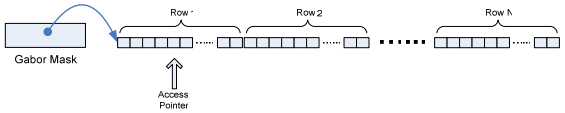
Figure 5. A 32-bit Fixed point number representation

3.3 Improving Array Access Efficiency using 1D Array

In general programming practice, storing data of an image / Gabor filter usually involves a logical 2D representation constructed by an array of pointers and a group of one dimensional pixel arrays. The problem with this scheme is that two pointer indirections are needed to access one image pixel, creating a heavy burden on the memory system. To reduce this overhead, one 1D array is used to store the whole image as illustrated in Figure 6. Rows of pixels are now packed sequentially, which means that nearby pixels can now be accessed with only one pointer indirection and one increment, thus, effectively reducing the burden on the memory system.



(a)



(b)

Figure 6 (a) Image pixels stored in a 2 structure (b) Image pixels stored in 1Dstructure

3.4 Improving access locality

In programming practices, multiple dimension array elements follow row-major ordering. This means that array elements adjacent to each other in memory differ in the second subscript instead of the first. For example, B(5,10) immediately follows B(5,9) in row-major ordering whereas B(5,10) would follow B(4,10) in column-major ordering. By interchanging the loop order and using suitable array formatting, image pixels brought into cache by a single memory read operation can be fully utilized before being evicted. This improvement essentially reduces the possibility of cache miss and page faults. Example of the memory access pattern is illustrated in Figure 7.

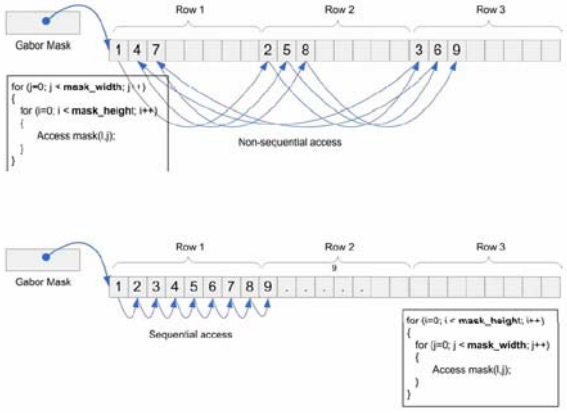


Figure 7 Memory access pattern. The original access pattern using column-major ordering (upper) and the optimized access pattern using row-major ordering (bottom)

3.5 Reducing Redundant Memory Access by Loop merging

In EBGM, there are a lot of filter convolutions consisting mainly of addition, multiplication and image element access. For brevity, the “complexity” of an algorithm can be simplified to two basic operations, namely Data Access (DACC) and Multiply and Add (MADD). Several important quantities and their values in are defined in Table 1.

Quantity	Value(s)
Mask size ($m \times m$)	m = {19, 29, 39,53,77}
Face image size ($N \times N$)	N = 128
NumOfMasks	80 (40 pairs)
NumOfNodes	25 (Landmark localization), 80 (Feature extraction)

Table 1 Quantities used in analysis

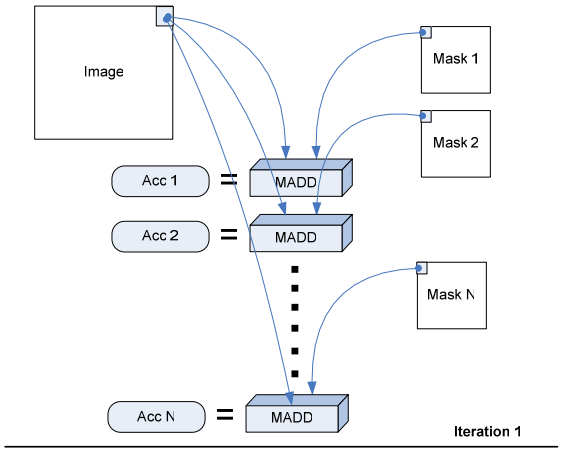
It is found that the most time consuming operation in EBGM is Gabor Jet extraction, involving finding the convolution responses to a set of Gabor masks (filter) at a specific location. The complexity of the Gabor jet extraction function (ExtractGaborJet) is linearly proportional to the number of Gabor masks used and the image size as depicted in Table 2. In the original implementation, each Gabor mask's element is loaded in to and out of the cache for convolution purpose, leading to a lot of page faults. By packing all the Gabor masks set into a single memory structure, all the filters can be convolved simultaneously in a contiguous memory space during each iteration cycle. This greatly reduces the total number of DACC operations. Table 3 shows the improved complexity of Gabor Jet extraction (ExtractGaborJetMultipleMasks) whose idea is depicted in Figure 8.

	General	Original
No. of DACC	$2 \times NumOfMasks \times m^2$	278240
No. of MADD	$NumOfMasks \times m^2$	139120

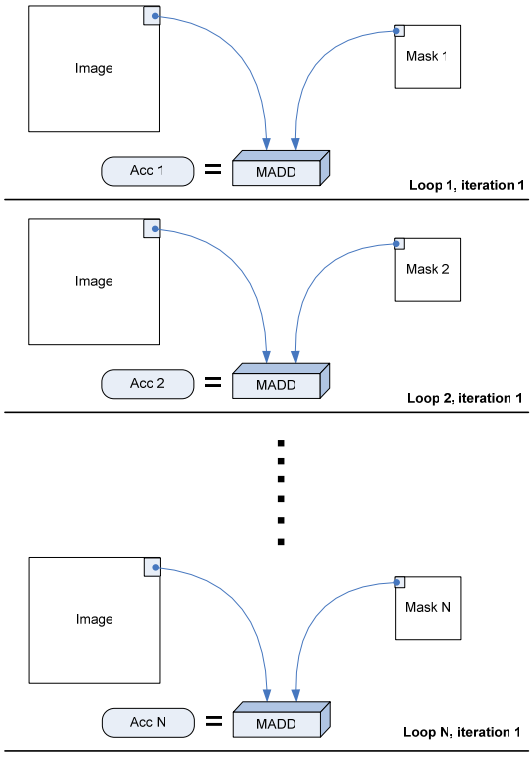
Table 2 Complexity for *ExtractGaborJet*

	General	Standard
No. of DACC	$\left(1 + \frac{1}{MaskSetSize}\right) \times NumOfMasks \times m^2$	147815
No. of MADD	$NumOfMasks \times m^2$	139120

Table 3 Complexity for ExtractGaborJetMultipleMasks



(a)



(b)

Figure 8. (a) Memory access pattern of original ExtractGaborJet function. (b) Memory access pattern of the ExtractGaborJetMultipleMasks using loop merging

3.6 Optimization of Trigonometric Function using Table look up

In a general purpose programming language, such as C, trigonometric functions are implemented using polynomial series. A large amount of floating point

multiplications are involved in the series expansion, which can result in poor execution time in mobile devices. To solve this problem, a table lookup technique is used. Values of trigonometric functions are pre-computed, stored and loaded into arrays when EBGM starts. Making use of the periodic nature and the relationship between trigonometric functions, only two tables are required to implement the sine, cosine, and tangent and arctangent functions. All calls to trigonometric functions are then modified to array accesses.

4. Experiment results

4.1 Experimental setup

Experiments were carried out to measure the effect of our optimization strategy. The execution time, space requirement and verification accuracy of the baseline and optimized system are compared. An Intel XScale PXA255 development board for mobile system is used for evaluation. Table 4 shows its configuration.

Processor	Intel XScale PXA255 400Mhz
Memory	64MB 100Mhz SDRAM
Storage	32MB flash ROM
OS	Embedded Linux (kernel version 2.4.19)
Compiler	arm-linux-gcc
Compile Options	-O2 –mtune=xscale

Table 4. Hardware specification of the mobile development board

4.2 Accuracy Test

To investigate the verification accuracy, the FERET verification testing protocol for face recognition [11] was used and four sets of face databases as summarized in Table 5 were investigated for the performance of our face authentication accuracy.

Probe category	Evaluation Task	Gallery size
FB	Facial expression	1196
dup1	Aging of subjects	1196
fc	Illumination	1196
dup2	Aging of subjects	1196

Table 5 Summary of probe categories used in the FERET test

In our experiments, the EBGM implementation of the CSU Face Identification Evaluation System 5.0 [10] was used as the baseline system. Figure 9 shows the Receiver Operating Characteristic (ROC) curves of the optimized code and the baseline system for different probe sets. The overlapping curves indicate that verification accuracies of both systems are essentially identical. These results show that our optimized face authentication does not incur any noticeable accuracy lost even on a mobile system.

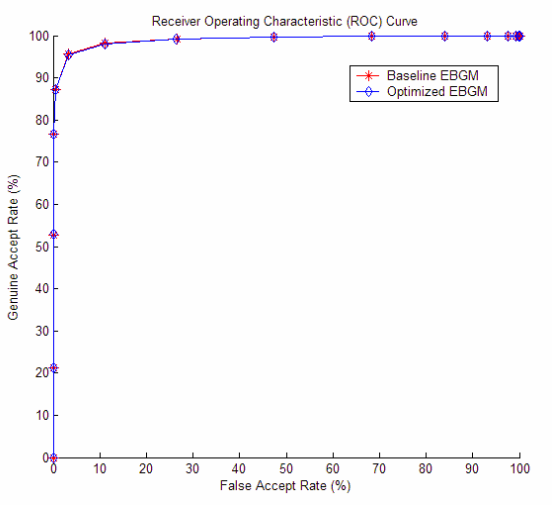


Figure 9 (a) ROC Curves of EBGM (**FB** probe set)

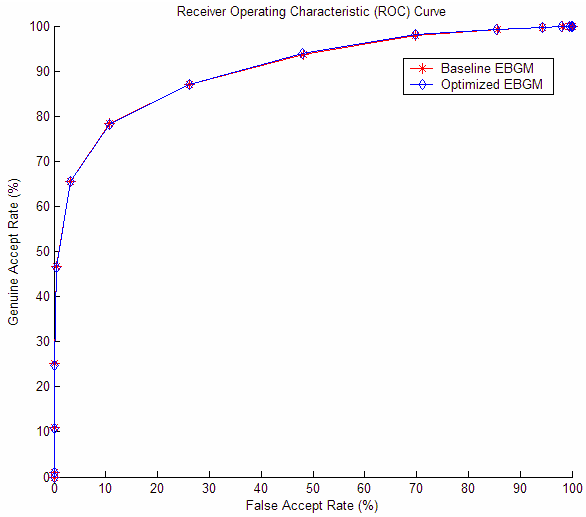


Figure 9 (b) ROC Curves of EBGM (**dup1** probe set)

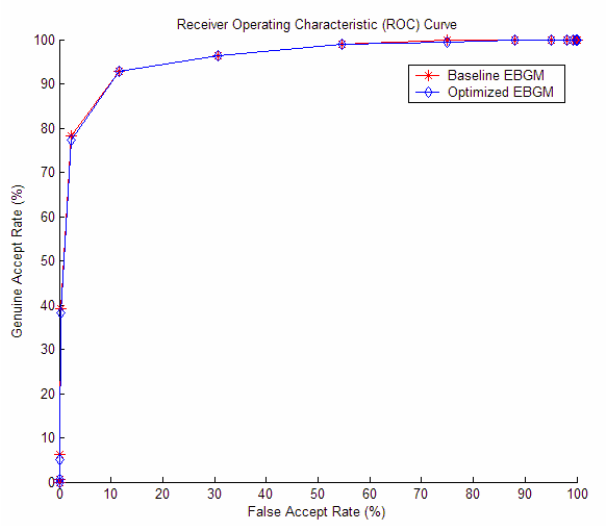


Figure 9 (c) ROC Curves of EBGM (**fc** probe set)

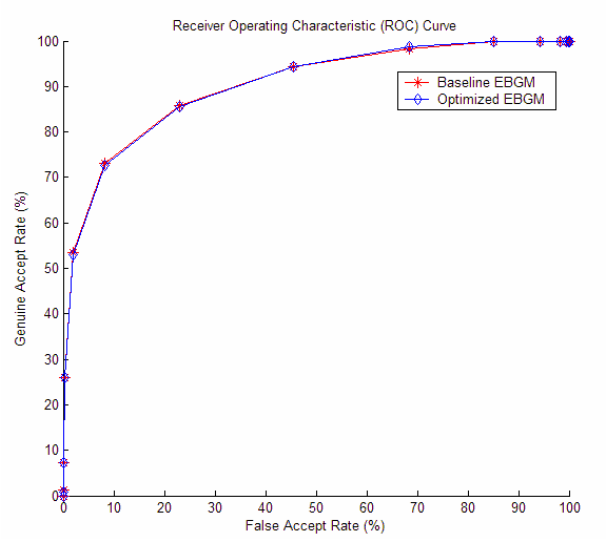


Figure 9 (d) ROC Curves of EBGM (**dup2** probe set)

4.3 Speed Test

To analyze the execution speed, we measured the mean execution time by averaging the total execution time against the number of face authentications (One face authentication: one face image matches with one face template). Table 6 summarizes the optimization techniques employed and their reductions in the execution time. Figures shown in bold indicate the particular stage that has been improved.

	Execution Time				
Technique (Section)	**Image Preprocessing**	**Landmark Localization**	**Feature Extraction**	**Template Matching**	**Total**
Original	8.335s	218.2s	324.8s	1.849s	553.2s
Fixed-point	**1.93s**	**15.07s**	**29.45s**	**0.2812s**	46.73s
Pre-computation	1.93s	**12.91s**	**20.97s**	0.2812s	36.09s
1D Array	**0.882s**	**7.026s**	**2.989s**	0.2970s	11.19s
Cache Policy	0.882s	**6.06s**	**2.78s**	**0.2263s**	9.948s
Loop Merging	0.882s	**5.87s**	**2.12s**	0.2263s	9.098s
Array Merging	0.882s	**5.85s**	**1.08**	0.2263s	8.043s
Table Lookup	**0.48s**	**0.52s**	**0.31s**	**0.0062s**	1.320s
Improvement (times)	**~17X**	**~420X**	**~1048X**	**~298X**	**~419X**

Table 6 Summary of optimization techniques and their related improvements

Although compiler, such as GNU GCC, can provide a certain extent of code optimization, generic optimization techniques do not cater for significant speed improvement to the EBGM algorithm if no domain specific optimization techniques are applied. As shown in Table 6, a single face authentication using the original EBGM algorithm runs for 553 seconds in a PXA255 mobile platform if only –O2 optimization flag is enabled in arm-linux-gcc compiler.

Referring to the above table, improvements to speed are mainly contributed by two methods. The fixed point arithmetic contributes to the major improvement by speeding up the execution time from 553s down to 46.73s (~10 times faster). In addition, the table lookup method speeds up the execution by ~7 times, and finally the reconstruction of the Gabor filters' data structure using 1D array helps to reduce the execution by ~ 3 folds. These results show that a thoughtful understanding of the domain specific requirements can significantly help an optimizing task.

5. Conclusion

We have developed a real-time face authentication system in a typical Intel XScale mobile platform as depicted in Figure 11. Our results enable the practical use of face authentication on mobile devices, especially mobile e-commerce system, which requires

secure identity checking and authorization. In addition, the generic optimization model and techniques developed in this paper can be easily adapted to other biometric applications which require real time performance on speed constrained platforms.

Figure 11. Our EBGM face authentication demo in a IPAQ 5400 PDA

6. References

[1] C. J. Liu and H. Wechsler, *"Evolutionary Pursuit and Its Application to Face Recognition,"* I9 Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 570-582, 2000.

[2] M. S. Bartlett, H. M. Lades, and T. J. Sejnowski, *"Independent Component Representations for Face Recognition,"* Proceedings of the SPIE, Vol 3299: Conference on Human Vision and Electronic Imaging 3, pp. 528-539, 1998.

[3] M. Turk, A. Pentland, *"Face Recognition Using Eigenface",* Journal of Cognitive Neuroscience, vol. 3, pp. 71-86, 1991.

[4] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, *"Eigenfaces vs. fisherfaces: Recognition using class specific linear projection,"* I9 Transaction on Pattern Analysis and Machine Intelligence, vol. 19, pp. 711-720, 1997.

[5] L. Wiskott, J. M.Fellous, N. Kruger, C. v. d. Malsburg, *"Face Recognition by Elastic Bunch Graph Matching,"* I9 Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pp. 775-779, 1997.

[6] P. S. Penev and J. J. Atick, *"Local Feature Analysis: A general statistical theory for object representation,"* Network: Computation in Neural Systems, vol. 7, pp. 477-500, 1996.

[7] " Intel XScale® Microarchitecture for the PXA255 Processor User Manual" Intel Corporation, March 2003.

[8] D. S. Bolme, Thesis: *"Elastic Bunch Graph Matching,"* Colorado State University, 2003.

[9] J. Zhang, Y. Yan, and M. Lades, "Face Recognition: Eigenface, Elastic Matching, and Neural Nets," Proceedings of the IEEE, No.9, vol. 85, pp. 1423-1435, 1997.

[10] D. S. Bolme, J. R. Beveridge, M. Teixeira, Bruce A. Draper, *"The CSU Face Identification Evaluation System: Its Purpose, Features, and Structure"*, ICVS 2003 .pp. 304-313

[11] S. A. Rizvi, P. J. Phillips, and H. Moon, *"The FERET Verification Testing Protocol for Face Recognition Algorithms,"* Proceedings of the International Conference on Automatic Face and Gesture Recognition, Nara, Japan, pp. 48-53, 1998.

[12] T. Y. Tang, Y. S. Moon, and K. C. Chan, "Efficient Implementation of Fingerprint Verification for Mobile Embedded Systems using Fixed-point Arithmetic," Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, pp. 821-825, 2004.

[13] K. H. Pun, Y. S. Moon, C.C. Tsang, C. T. Chow, and S. M. Chan, "A Face Recognition Embedded System," Proceedings of Biometric Technology for Human Identification II, SPIE Defense and Security Symposium, Florida, USA, pp. 390-397, 2005.