http://dx.doi.org/10.5755/j01.eee.20.6.7287 ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392–1215, VOL. 20, NO. 6, 2014

Fingerprint Pre-processing on ARM and DSP Platforms

M. Gok[1], S. Gorgunoglu[1], I. Muharrem Orak[1]
[1]*Department of Computer Engineering, Karabuk University, Karabuk, Turkey*
gokmehmet@outlook.com

***Abstract*—In minutiae based fingerprint analysis, fingerprint image is pre-processed before extracting features. The pre-processing is carried out to obtain more accurate minutiae points. Implementing fingerprint programs on embedded systems can be considered as important especially for real time standalone applications. Reducing the pre-processing time is important for identification and verification in real time embedded systems. In this study, pre-processing of minutiae based fingerprint system is implemented on two different platforms: Texas Instruments Sitara AM3359 which is a single board computer and OMAP-L138 which is a development kit. OMAP-L138 is a low power application processor based on ARM9 and C674x DSP cores. AM3359 is microprocessor unit based on ARM Cortex-A8 core. Fingerprint pre-processing algorithms are implemented using C/C++ compiler and tested on three different cores: ARM9, DSP and ARM Cortex-A8. The execution times are compared with each other. The results show that using DSP core, execution time is substantially improved.**

***Index Terms*—Fingerprint recognition, data pre-processing, performance analysis, digital signal processors.**

I. INTRODUCTION

Fingerprint, iris, retina, signature, face, DNA etc. are used in biometric system. These characteristics are sometimes combined together to improve the security of the systems. Finger vein also is used for access control [1]. In another study, to enhance the security of the computer access, colours are considered [2]. Fingerprint recognition and verification systems shows better result comparing to other biometric systems.

For a minutiae based fingerprint recognition system, before extraction minutiae, the pre-processing steps shown in Fig.1 are applied. By applying these steps, more accurate features are extracted from the fingerprint image taken by a sensor. Pre-processing time consumes the most of the processing time in minutiae based fingerprint recognition system. Especially for a real time embedded system requiring fast response time, pre-processing time has to be reduced as much as possible.

In this study, a GUI (Graphical User Interface) fingerprint pre-processing application designed on a PC (Personal Computer) and then ported to evaluation platforms using cross compiler and design environments. The execution times of algorithms are evaluated on Sitara AM3359 Cortex-A8 core, OMAP-L138 ARM9 core and C6748 DSP core.

Manuscript received October 7, 2013; accepted January 15, 2014.

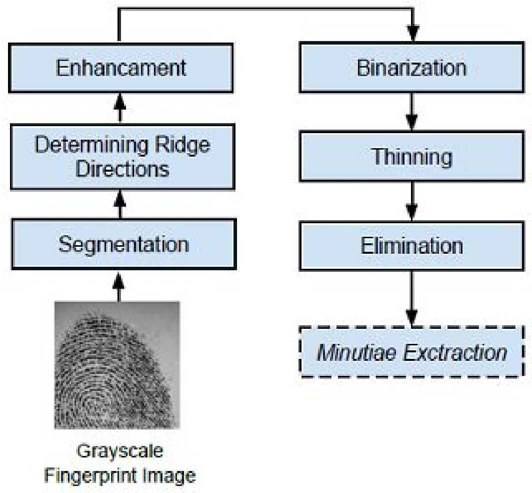


Fig. 1. The pre-processing algorithms used in minutiae based fingerprint analysis.

II. PRE-PROCESSING ALGORITHMS

A. Segmentation

Segmentation process is used to separate the fingerprint image from the background image [3]. In the fingerprint image, there may be some distorted areas or the regions where the ridges and valleys of the fingerprint are not clear. These unwanted regions are segmented by the algorithm. In the segmentation based on variance value, fingerprint image is divided into *wxw sized* blocks. The variation of each block is calculated. If a variance of a block is less than a specified threshold, this block is signed as a background image (1) which does not contain fingerprint information [4].

These steps are carried out by using following equations where *m(i, j)* is the average gray scale value of *wxw* sized block (2) and *v(i, j)* is the variance of the block (3).

$$m(i,j) = \frac{1}{w*w}\sum_{u=i-w/2}^{i+w/2}\sum_{v=j-w/2}^{j+w/2} I(u,v), \qquad (1)$$

$$v(i,j) = \frac{1}{w*w}\sum_{u=i-w/2}^{i+w/2}\sum_{v=j-w/2}^{j+w/2} (I(u,v)-m(i,j))^2, \qquad (2)$$

$$C = \begin{cases} 1 & if \quad v(i,j) > Threshold, \\ 0 & otherwise. \end{cases} \qquad (3)$$

B. Determining Ridge Direction

The directions of the ridges on fingerprint are used in several purposes such as to enhance the quality of the fingerprint image and to classify the fingerprint. The following steps are applied to determine the ridge direction [5]. Fingerprint is divided into *wxw* sized blocks. The gradient values in x and y directions of each pixel in the block are calculated by sobel operator of ($\partial_x(i,j)$, $\partial_y(i,j)$ as given in (4) and (5):

$$\begin{cases} \partial_x(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \\ \partial_y(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \end{cases} \tag{4}$$

The direction of *wxw* sized block centred in (i,j) is calculated as given in (6):

$$V_y(i,j) = \sum_{u=i-W/2}^{i+W/2} \sum_{v=j-W/2}^{j+W/2} 2\partial_x(u,v)\partial_y(u,v), \tag{5}$$

$$V_x(i,j) = \sum_{u=i-W/2}^{i+W/2} \sum_{v=j-W/2}^{j+W/2} \partial^2{}_x(u,v) - \partial^2{}_y(u,v), \tag{6}$$

$$^{-}(i,j) = \frac{1}{2}\tan^{-1}\frac{V_y(i,j)}{V_x(i,j)}, \tag{7}$$

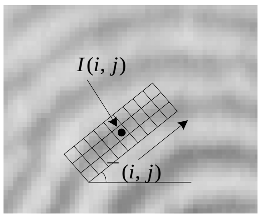
where $^{-}(i,j)$ gives the angular direction of the block (7) centred in (i, j).

C. Enhancement Algorithms

Enhancement is the elimination process of fingerprint from noises. After determination of the ridge direction, the mask of (3 × 9) shown in Fig. 2 is applied to fingerprint image taking the direction of ridge into consideration. By means of this process, tiny disconnections are repaired and also the ridge is smoothed [6].

	-4	-3	-2	-1	0	1	2	3	4
-1	0	1	4	9	16	9	4	1	0
0	25	16	9	4	1	4	9	16	25
1	0	1	4	9	16	9	4	1	0

a)



b)

Fig. 2. Enhancement mask and its application.

Enhancement algorithm consists of following steps:

Transformed coordinates of *I(i,j)* pixel are calculated by following equations:

$$\begin{cases} j' = j + (u.\cos^{-} + v.\sin^{-}), \\ i' = i + (-u.\sin^{-} + v.\cos^{-}), \end{cases} \tag{8}$$

where u and v take (-4, -3, …, 3, 4) and (-1, 0, 1) values accordingly. The value of the $I(i,j)$ pixel centred in the mask is obtained from (10):

$$K = \sum_{u=-1}^{1} \sum_{v=-4}^{4} W(u,v), \tag{9}$$

$$I(i,j) = \frac{1}{K} \sum_{u=-1}^{1} \sum_{v=-4}^{4} W(u,v) I(i', j'). \tag{10}$$

D. Binarization

Binarization is the transformation process of image pixels into black or white scale. In this stage, the average value of the local block is taken as threshold for binarization. This process is done by applying following steps.

Fingerprint is divided into *wxw* sized blocks where *w* is taken as 9. The average value of gray scale value for block centred in (i,j) is calculated (11)

$$I_{Local\ mean}(i,j) = \frac{1}{w*w} \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} I(u,v). \tag{11}$$

If $I(i,j) > I_{Local\ mean}(i,j)$, then pixel is signed as white and otherwise as black. This equation can be stated as given in (12):

$$C = \begin{cases} 255\ (white),\ if\ \ I(i,j) > I_{Local\ mean}(\text{i,j}), \\ 0\ (black), \qquad\qquad\qquad otherwise. \end{cases} \tag{12}$$

E. Thinning Fingerprint

Thinning process is to express the binarized fingerprint in terms of a single pixel thickness. Thinning process consists of four stages [7], [8]. Firstly, the targeted pixel has to be black. If the pixel is white this step is skipped. Secondly, vertical and horizontal neighbours of the targeted pixel have to be white. Thirdly, at least one neighbour of the pixel has to be black. Finally, while the targeted pixel is to be removed, if connection is breaking down, the pixel has not to be removed.

F. Elimination Algorithm

If enhanced fingerprint image is studied carefully, on fingerprint ridges, disconnection, tiny bifurcation points and small lines in few pixels length can be found. These have to be removed by an elimination algorithm [9], [10].

III. EVALUATION PLATFORMS

A. AM3359 Sitara Processor Platform

The first test platform used in this study is a SBC (single board computer) called BeagleBone Black. This SBC is based on Sitara AM3359 MPU. AM3359 is manufactured by Texas Instruments, has an ARM Cortex-A8 32-bit RISC core running at speed of 1 Ghz. Sitara processors are designed for high performance, feature-rich and low-power industrial and consumer applications [11]. In this study, HDMI output of the board is converted to VGA signal using external converter hardware in order to display pre-processor application GUI on a monitor.

The operating system running on BeagleBone Black is *Angstrom Linux 2013.09.05* distribution build for ARM Cortex-A8 architecture. Windows manager is disabled to save system resources so the application uses Linux frame buffer mechanism to display GUI on monitor.

Design of pre-processor application is realized on a PC running Linux Mint 15 operating system. This application is built with Qt 4.8.5 cross platform GUI library using ×86. For PC target of our application, pre-built Qt 4.8.4 library provided in Linux Mint is used with GCC 4.7.3 compiler. For ARM target of our application, Qt 4.8.5 source code is configured and built for ARM Cortex-A8 architecture using GCC version 4.6.3 (Sourcery CodeBench Lite 2012.03-57). ARM version of the application is compiled and built on PC and deployed to target BeagleBone through Ethernet. For this reason build time is reduced.

B. *OMAP-L138 Processor Platform*

Our second test platform is Zoom OMAP-L138 eXperimenter kit. This kit has OMAP-L138 SOM (System On Module). SOM has 8 Mbytes of NOR Flash and 128 Mbytes of Mobile DDR SDRAM with OMAP-L138 processor.

The OMAP-L138 C6000 DSP+ARM processor is a low-power applications processor based on high performance TMS320C674x DSP core and an ARM926EJ-S core [12]. It's up to 456MHz with high performance and low price.

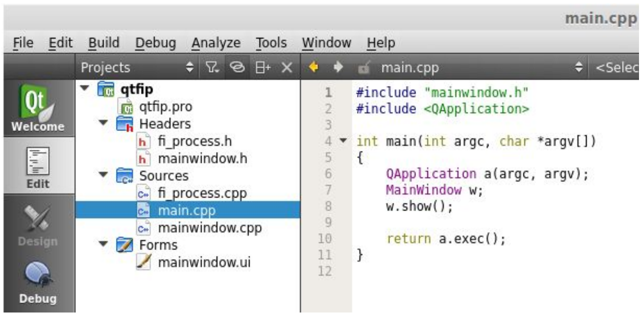
The operating system running on OMAP-L138 kit is *Arago Linux 2011.06* provided within Texas Instruments Digital Video Software Development Kit (DVSDK). This kit is a complete development framework including Cross Compiler GCC 4.3.3, pre-built Qt 4.6.3 library and DSP software libraries for ARM9 architecture.

IV. DEVELOPMENT OF PRE-PROCESSOR APPLICATION

Development of our application is realized using Qt Creator IDE. All the pre-processing routines are coded from scratch without use of any third party image processing library.

A. *Project Source of Pre-Processor Application*

In the project tree (Fig. 4) the *qtfip.pro* is the project file including build information for application. fi_process.h includes function prototypes for pre-processing functions. mainwindow.h includes definitions for main window of the application. fiprocess.cpp includes fingerprint pre-process function implementations. main.cpp includes application main entry. mainwindow.cpp has the routines for main window of the application.



```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Fig. 3. Project tree of the pre-processor application.

The application is built for AM3359 and OMAP-L138 ARM9 core using this project structure and appropriate libraries and tool chains. For DSP implementation of the algorithms, an extra tool is used. This tool provides access to DSP core from ARM side of OMAP-L138 processor.

B. *C6Run and DSP Side Programming*

The C6Run tool moves DSP development into the domain of the ARM/Linux programmer by masking many of the details of programming the DSP. With C6Run, the user can move pieces of an application from the ARM core to the DSP core by simply rebuilding code. This tool enables code executing on the DSP to appear as just another process in the ARM operating system [13].

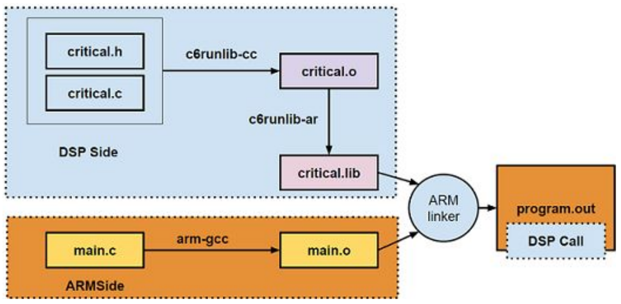


Fig. 4. C6RunLib build mechanism.

In this study, C6RunLib tools are used to run algorithm on DSP side. The c6runlib tools consist of a compiler (c6runlib-cc) and an archiver script (c6runlib-ar). The compiler script allows the developer to compile C code to C6000 object file format. The archiver tool is used to create a library from one or more object files. The c6runlib archiver tool takes the C6000 object files to produce a static ARM library. As shown in Fig. 4, this library is then linked to ARM-side object files to produce an ARM executable.

V. EXPERIMENTAL STUDIES

For the test purposes, fingerprint test images are taken from the FVC2004 fingerprint database. Test set includes sampled DB1_A, and DB4_A images. Time measurement is done in terms of milisecond (ms) using Qt timer API. ARM and DSP execution times of the algorithms are given in Table I and Table II.

TABLE I. EXECUTION TIME OF PRE-PROCESSING ALGORITHMS ON DIFFERENT CORES FOR 288 × 384 SIZED IMAGE.

Pre-processing algorithms	**Execution Time (ms)**			
	AM3359 ARM Cortex A8 (1 GHz)	**OMAP-L138 ARM9 (456 MHz)**	**OMAP-L138 C674x DSP (456 MHz)**	**Speed up Ratio**
Segmentation	13	299	12	1.083
Determining ridge direction	320	4762	1014	0.316
Enhancement	682	12185	459	1.486
Binarization	1587	12933	464	3.420
Thinning	196	1603	125	1.568
Elimination	544	5723	243	2.239
Total	3342	37505	2317	1.442
Image size (288 × 384), Number of pixels(110592), Local block size w = 24, DB4_A (1_1.tif)				

It can be seen from Table I and Table II that, pre-processing done with DSP has achieved better performance although running at slower clock speed. Only *determining ridge direction* step is slower than ARM Cortex A8 Core.

TABLE II. EXECUTION TIME OF PRE-PROCESSING ALGORITHMS ONDIFFERENT CORES FOR 640 × 480 SIZED IMAGE.

Pre-processing algorithms	**Execution Time (ms)**			
	AM3359 ARM Cortex A8 (1 GHz)	**OMAP-L138 ARM9 (456 MHz)**	**OMAP-L138 C674x DSP (456 MHz)**	**Speed up Ratio**
Segmentation	36	695	28	1.286
Determining ridge direction	679	9257	1461	0.465
Enhancement	1735	29674	1186	1.463
Binarization	3259	19251	1010	3.227
Thinning	338	2006	238	1.420
Elimination	1585	16855	594	2.668
Total	7632	77738	4517	1.690
Image size (640 × 480), Number of pixels(307200), Local block size w = 24 DB1_A (1_1.tif)				

Therefore DSP execution time of this step is slower than Cortex-A8 core. Other than this step, ARM9 core shows worse performance in terms of execution time. This can be clearly seen on Fig. 5.

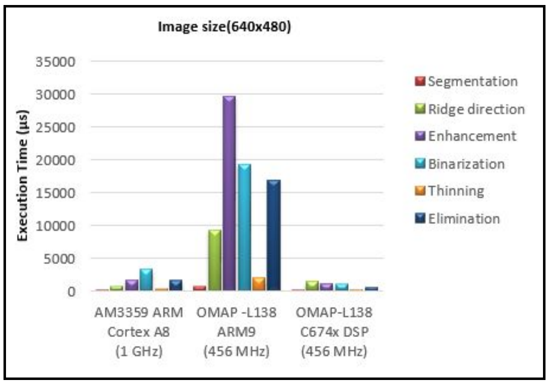


Fig. 5. Execution time on different cores using 640 × 480 sized image.

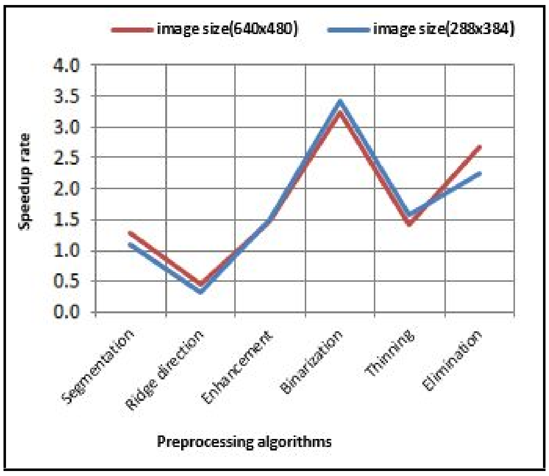


Fig. 6. The speedup ratio of pre-processing algorithms.

The speedup ratio between selected platforms (i.e. ARM Cortex A8 and L138 C674x DSP core) are calculated for both sampled data on Table I and Table II. These ratios are shown together also on Fig. 6. It is obvious that with the increase of image sizes, execution times also increase linearly. Therefore the size of images does not have impact on the speed-up ratio. It can also been seen from Table I and Table II that average speedup ratio for pre-processing times are 1.442 and 1.690 accordingly.

VI. CONCLUSIONS

In this study, pre-processing of minutiae based fingerprint system is realised on two different processors core. Execution time of each pre-processing algorithm on ARM and DSP microprocessors are computed and also comparison is done by speed-up ratio. These values show which algorithm can be executed effectively in which microprocessor. It has been shown that pre-processing algorithms for fingerprint can be implemented in embedded systems. Although the image dimensions in FVC2004 database are rather large for an embedded system, test results show that both ARM Cortex-A8 and DSP cores can be used in an embedded fingerprint verification and recognition system. ARM9 core of OMAP-L138 is not suitable for operations requiring floating point operations. It is suitable for running embedded operating system and user interface tasks. In future work, minutiae extraction and fingerprint matching algorithms will be ported for evaluation platforms. A fingerprint database will be created and tests will be done for real world conditions. A capacitive fingerprint sensor like FPC1011F will be interfaced to the microprocessors to complete development platform. This sensor provides a high quality fingerprint image.

REFERENCES

[1] A. Venckauskas, N. Morkevicius, K. Kulikauskas, “Study of finger vein authentication algorithms for physical access control”, *Elektronika ir Elektrotechnika*, vol. 121, no. 5, pp. 101–104, 2012.
[2] F. Yucel, O. Oral, N. Caglayan, M. Tecimen, S. Kocak, E. Yuce, “Design and implementation of a personal computer system using color detection”, *Elektronika ir Elektrotechnika*, vol. 115, no. 9, pp. 97–100, 2011.
[3] S. Gorgunoglu, A. Cavusoglu, “A fast and simple algorithm for fingerprint segmentation”, *Teknoloji*, vol. 11, no. 2, pp. 87–92, 2008.
[4] B. M. Mehtre, B. Chatterjee, “Segmentation of fingerprint images – a composite method”, *Pattern Recognition*, vol. 22, no. 4, pp. 381–385, 1989. [Online]. Available: http://dx.doi.org/10.1016/0031-3203(89)90047-2
[5] L. Hong, Y. Wan, A. Jain, “Fingerprint image enhancement: algorithm and performance evaluation”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777–789, 1998. [Online]. Available: http://dx.doi.org/10.1109/34.709565
[6] A. Cavusoglu, S. Gorgunoglu, “A fast fingerprint image enhancement algorithm using a parabolic mask”, *Computers & Electrical Engineering*, vol. 34, no. 3, pp. 250–256, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2006.11.006
[7] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, San Diego: California Technical Publishing, pp. 436–442, 1999.
[8] V. Espinosa-Duro, “Fingerprints thinning algorithm”, *Aerospace and Electronic Systems Magazine IEEE*, vol. 18, no. 9, pp. 28–30, 2003. [Online]. Available: http://dx.doi.org/10.1109/MAES.2003.1232157
[9] M. Tico, P. Kuosmanen, “An algorithm for fingerprint image postprocessing”, in *34th Asilomar Conf. Signals, Systems and Computers*, vol. 2, 2000, pp. 1735–1739.
[10] Q. Xiao, H. Raafat, “A combined statistical and structural approach for fingerprint”, *IEEE Int. Conf. Image Postprocessing, Systems, Man and Cybernetics*, 1990, pp. 331–335.
[11] *Sitara AM335x ARM Data Manual*, Texas Instruments Incorporated, 12500 TI Boulevard Dallas, 2013.
[12] A. Wang, F. Pan, Y. Li, R. Tao, “The design of power quality detecting system based on OMAP-L138”, in *IEEE 13th Workshop on Control and Modelling for Power Electronics (COMPEL 2012)*, 2012, pp. 1–4.
[13] *C6Run DSP Software Development Tool White Paper*. Texas Instruments Incorporated, 12500 TI Boulevard Dallas, 2013.