Ilorin Journal of Science
Volume 1, Number 2, 2014, pp. 245 – 263 (Printed in Nigeria)
ISSN: 2408 – 4840 © 2014 Faculty of Physical Sciences, University of Ilorin
https://doi.org/10.54908/iljs.2014.01.02.002

ILORIN
JOURNAL OF SCEINCE

ILJS-14-028

A Review of Algorithms for Fingerprint Image Acquisition, Preprocessing and Minutiae Extraction

Adewole, K. S., Jimoh, R. G.* and Abikoye, O. C.

Department of Computer Science, University of Ilorin, Ilorin, Nigeria

Abstract

Biometric recognition distinguishes between individuals using physical, chemical or behavioral attributes of the person. These attributes are called biometric identifiers or traits, and include fingerprint, palmprint, iris, face, voice, signature, gaint, and DNA among others. Fingerprint recognition is one of the oldest and most reliable biometric used for personal identification. Fingerprint has come a long way from tedious manual fingerprint matching. The ancient procedure of matching fingerprints manually was extremely cumbersome and time-consuming and required skilled personnel. In this paper, a review of algorithms for the various stages involved in fingerprint recognition such as fingerprint image acquisition, segmentation, normalization, ridge orientation estimation, ridge frequency, Gabor filtering, binarization, thinning, minutiae extraction, template generation, and template matching is presented. It was established that minutiae features of a person fingerprint truly make fingerprint of individual to be unique.

Keywords: Biometric, Fingerprint, Algorithm, Filtering, Minutiae

1. Introduction

Fingerprint recognition is one of the most popular biometric techniques used in automatic personal verification and identification (Naser, 2011). Most fingerprint verifications systems use minutiae point matching. The minutiae points are the points present in fingerprint images where the fingerprint ridges either end or splits up into two new ridges. There are two main approaches to minutia detection in fingerprint images: binary detection and direct grayscale detection (Erikson, 2001). This paper considers the first approach to minutiae detection and present extensive studies in this area. There are several stages involved in extracting minutiae points from binarized fingerprint image. These stages include fingerprint image acquisition, segmentation, normalization,

*Corresponding Author: Jimoh, R. G.
Email: jimoh_rasheed@yahoo.com

ridge orientation estimation, ridge frequency, Gabor filtering, binarization, thinning, minutiae extraction, template generation, and template matching. The minutiae are defined as the pattern created and the uniqueness of how ridges end, split and join, or appear as a simple dot. The minutiae consists of bifurcations, ridge dots, ridge endings and enclosures, to ensure further uniqueness, the minutiae are further broken down into sub minutiae such as pores, crossovers, deltas. The pores are tiny depressions within the ridge on a fingerprint; the crossover creates an X pattern within the ridge of a fingerprint and deltas create a triangle shaped pattern within the ridge of a fingerprint. Identification in a fingerprint technology exists when an individual fingerprint is compared against a known source called the fingerprint template (Chirillo & Scott, 2007).

Crossing Number (CN) approach is the most commonly used method of minutiae feature extraction from binarized fingerprint image (Roli *et al.*, 2011; Raymond, 2003; Anil *et al.*, 2008). This method extracts the ridge endings and bifurcations from the skeleton image. The minutiae features are extracted by scanning the local neighborhood of each ridge pixel in the image using a 3 x 3 window (Sunny, 2012). The CN value is then computed, which is defined as half the sum of the differences between metric pairs of adjacent pixels as shown in equation 1.1:

$$CN = 0.5\sum_{i=1}^{8}|P_i - P_{i+1}|, \qquad P_9 = P_1 \tag{1.1}$$

where P_i is the pixel value in the neighborhood of P. The neighborhood of a pixel P are scanned in anti-clockwise direction as shown in Table 1.1

Table 1.1: Ridge pixel local neighborhood

P_4	P_3	P_2
P_5	P	P_1
P_6	P_7	P_8

Using the properties of the CN in Table 1.2, the ridge pixel can be classified as a ridge ending if it has only 1 neighboring ridge pixel in the widow or as bifurcation if it has three neighboring ridge pixels or non-minutiae point.

Table 1.2: Properties of Crossing Number

CN	Property
0	Isolated point
1	Ridge ending point
2	Continuing ridge point
3	Bifurcation point
4	Crossing point

Each of the extracted minutiae points has *x* and *y* coordinates, orientation of the associated ridge segment (θ), and the type of minutiae.

2. Materials and Methods

Fig. 1 shows the conceptual diagram presented in this paper to guide the review process. In this section, the various components in the framework are discussed and their algorithms presented.

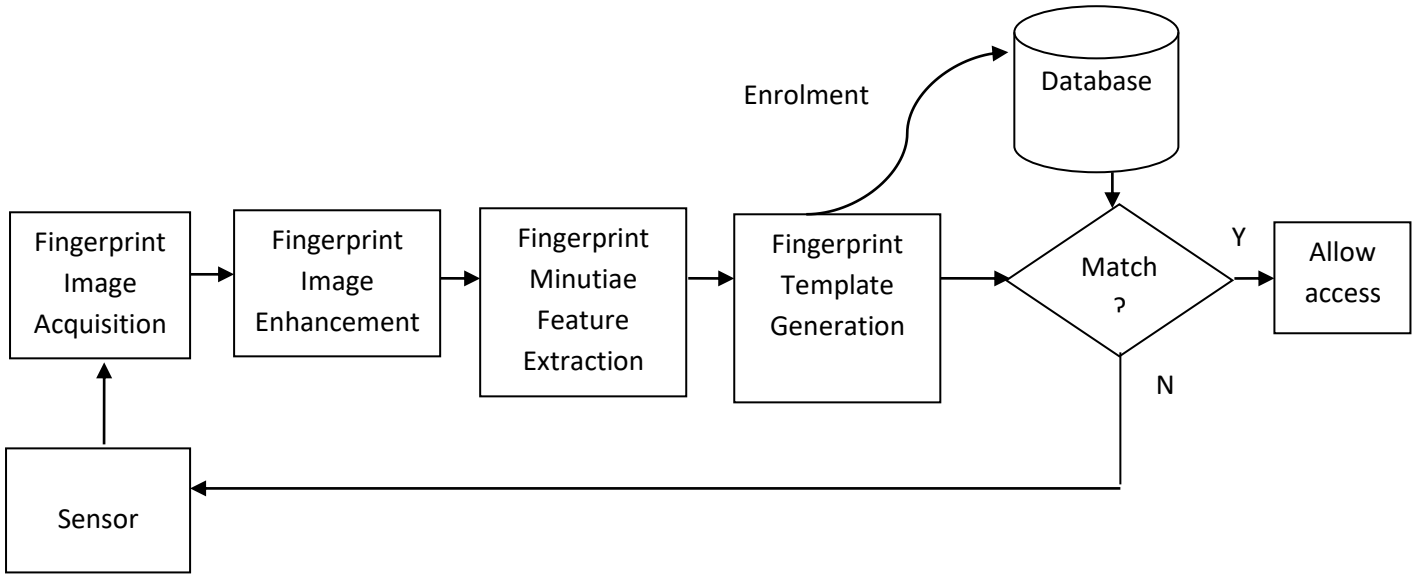

Fig. 1: Conceptual diagram of the system

Fingerprint Sensors

Fingerprint sensor is an electronic device used to capture a digital image of the fingerprint pattern (Cappelli *et al.*, 2006). The captured image is called a live scan. This live scan is digitally processed to create a biometric template (a collection of extracted minutiae points) which is stored and used for matching. Different types of fingerprint scanners include capacitive sensor, optical sensor, thermal sensor, pressure sensor, and RF sensor among others (Sharat, 2005; http://360biometrics.com/faq/fingerprint_scanners.php). Figure 2 shows a SecuGen fingerprint optical sensor.

Fig 2: Secugen fingerprint sensor

Fingerprint Image Acquisition

A fingerprint is a unique pattern of ridges and valleys on the surface of a finger of an individual. This fingerprint is captured using any of the fingerprint sensors mentioned in the previous section. A ridge on the fingerprint is defined as a single curved segment, and a valley is the region between two adjacent ridges. Minutiae points are the local ridge discontinuities, which are of two major types: ridge endings and bifurcations. Previous research has shown that a good quality fingerprint image has around 40 to 100 minutiae (Roli, Priti & Punam, 2011). In the fingerprint image acquisition module, fingerprint scanner is used to capture the fingerprint image of the user. The algorithm for this module is as follows:

FingerPrintImageAqc()

{

Step 1: Connect fingerprint optical scanner through the USB port

Step 2: Place fingerprint on the scanner

Step 3: Scan the fingerprint image (*I*) at 500 dots per inch (dpi)

Step 4: Save the fingerprint image (*I*) on the computer system

Step 5: Stop

}

Fingerprint Image Enhancement

This module contains several sub-modules in order to enhance the quality of the scanned fingerprint image in FingerPrintImageAqc() module discussed above. The sub-modules as well as their algorithms are described in this section:

Segmentation() module: This module is used to separate the foreground of the fingerprint image from its background. The foreground regions correspond to the clear fingerprint area containing the ridges and valleys corresponding to the area of interest. The background corresponds to the regions outside the borders of the fingerprint area and these regions do not contain any valid fingerprint information (Raymond, 2003).

Normalization() module: The normalization module is used to standardize the intensity values in an image by adjusting the range of grey-level values so that it lies within a desired range of values.

RidgeOrientation() module: Ridge orientation field of a fingerprint image is the local orientation of the ridges contained in the fingerprint. Calculating the estimate of this orientation is a very important step as the subsequent Gabor filtering stage relies on the local orientation in order to effectively improve the fingerprint image.

RidgeFrequency() module: This module is used to extract another important parameter that is used by Gabor filter. This parameter represents the local frequency of the ridges in a fingerprint image.

GaborFiltering() module: The computed ridge orientation and ridge frequency parameters are used by this module for Gabor filter. Gabor filter has frequency-selective and orientation-selective properties. The Gabor filter is applied to the fingerprint image by spatially convolving the image with the filter (Raymond, 2003).

Binarization() module: Binarization is the process of converting a gray-scale image to binary image. In a gray-scale fingerprint image, a pixel can take on 256 different intensity levels. The threshold value is used to convert grayscale image to binary. According to Ghazali (2005), the various techniques that can be used to convert gray-scale image to binary image are Global thresholding, Regional average thresholding, Histogram based thresholding, and Niblack binarization. However, Regional Average Thresholding (RAT) method was

described in this paper because, it preserves useful information in the fingerprint image unlike global thresholding technique that can corrupt the fingerprint image.

Thinning() module: Thinning is a morphological operation that successively erodes away the foreground pixels until they are one pixel wide (Raymond, 2003; Sozan, 2011). The application of the thinning algorithm to a fingerprint image preserves the connectivity of the ridge structures while forming a skeletonized version of the binary image. This skeleton image is then used in the subsequent extraction of minutiae. The algorithms for these modules are as follows:

Segmentation(fingerprint_image *I*)

{

Step 1: Set a global threshold (*T*) for the segmentation operation

Step 2: Divide the fingerprint image (*I*) into blocks of size *W x W*

Step 3: Compute the mean grey-level value *M(k)* for each block *k* in the image (*I*) i.e

For k = 0 to *N* where *N* is the number of blocks

For i = 0 to W-1

For j = 0 to W-1

Sum = Sum + *I(i, j)* where *I(i, j)* is the grey-level value at pixel *(i, j)*

End j

End i

M(k) = Sum/W where *M(k)* is the mean grey-value for block *k*

End k

Step 4: Obtain the grey-scale variance *V(k)* for each block *k* i.e

For i = 0 to W-1

For j = 0 to W-1

$V(k) = V(k) + (1/W^2 * (I(i, j) - M(k))^2)$ where $I(i, j)$ is the grey-level value at pixel (i, j)

end j

end i

If($V(k) < T$) then

Assign block *k* in image (*I*) as a background

k = k +1

goto Step 4 until no more block to check

else

Assign block *k* in image (*I)* as a foreground

k = k + 1

goto Step 4 until no more block to check

endif

Step 5: Stop

}

Normalization(segmented_image *I*)

{

Step 1: Obtain the mean value (*M*) of the image (*I*) i.e

For i = 0 to H-1 where H is the number of rows in the fingerprint image (*I*)

For j = 0 to W-1 where W is the number of columns in the fingerprint image (*I*)

Sum = Sum + $I(i, j)$ where $I(i, j)$ is the grey-level value at pixel (i, j)

End j

End i

M = Sum/Total where Total is the size of the segmented fingerprint image

Step 2: Obtain the variance (V) of the image (*I*) i.e

For i = 0 to H-1

For j = 0 to W-1

$D = D + (I(i,j) - M)^2$ where D is the deviation value

End j

End i

V = sqrt(*D*/Total)

Step 3: Compute the desired mean value M_0 and variance V_0 respectively

Step 4: Obtain the normalized grey-level value $N(i,j)$ at pixel (i,j) in the segmented fingerprint image (I) such that:

For i = 0 to H-1

For j = 0 to W-1 where W is the width of the entire fingerprint image (I)

If($I(i,j) > M$) then

$N(i,j) = M_0 + \text{sqrt}(V_0 * (I(i,j) - M)^2)$

Else

$N(i,j) = M_0 - \text{sqrt}(V_0 * (I(i,j) - M)^2)$

Endif

End j

End i

Step 5: Stop

}

RidgeOrientation(Normalized_image *N*)

{

Step 1: Divide the normalized fingerprint image (N) into a block of size $W \times W$

Step 2: For each pixel in the block *k*, compute the gradients $\partial x(i,j)$ and $\partial y(i,j)$ which are the gradient magnitudes in the *x* and *y* directions, respectively. The horizontal Sobel operator is used to compute $\partial x(i,j)$ and the vertical Sobel operator is used to compute $\partial y(i,j)$.

Step 3: Obtain the local orientation of the block centered at pixel (*i, j*) as follows:

$$Vx(i,j) = \sum_{u=i-(w/2)}^{i+(w/2)} \sum_{v=j-(w/2)}^{j+(w/2)} 2\partial x(u,v)\partial y(u,v) \tag{2.1}$$

$$Vy(i,j) = \sum_{u=i-(w/2)}^{i+(w/2)} \sum_{v=j-(w/2)}^{j+(w/2)} \partial^2 x(u,v)\partial^2 y(u,v) \tag{2.2}$$

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\frac{Vy(i,j)}{Vx(i,j)} \tag{2.3}$$

Where $\theta(i,j)$ is the least square estimate of the local orientation at the block centered at pixel (*i, j*).

Step 4: The orientation field is then smooth in a local neighborhood using a Gaussian filter. The orientation is firstly converted into a continuous vector field, which is defined as:

$$\Phi x(i,j) = \cos(2\theta(i,j)) \tag{2.4}$$

$$\Phi y(i,j) = \sin(2\theta(i,j)) \tag{2.5}$$

Where Φ_x and Φ_y are the x and y components of the vector field, respectively. Gaussian smoothing is then performed after the vector field is computed. This can be obtained as follows:

$$\Phi' x(i,j) = \sum_{u=\frac{-w\Phi}{2}}^{\frac{w\Phi}{2}} \sum_{v=\frac{-w\Phi}{2}}^{\frac{w\Phi}{2}} G(u,v)\Phi x(i-uw, j-vw) \tag{2.6}$$

$$\Phi' y(i,j) = \sum_{u=\frac{-w\Phi}{2}}^{\frac{w\Phi}{2}} \sum_{v=\frac{-w\Phi}{2}}^{\frac{w\Phi}{2}} G(u,v)\Phi y(i-uw, j-vw) \quad (2.7)$$

Where G is a Gaussian low-pass filter of size $w\phi$ x $w\phi$.

Step 5: The final step is to obtain the smooth orientation field O at pixel (i, j). This can be obtained as follows:

$$O(i,j) = \frac{1}{2}\tan^{-1}\frac{\Phi' y(i,j)}{\Phi' x(i,j)} \quad (2.8)$$

Step 6: Stop

}

RidgeFrequency(Normalized_image *N*)

{

Step 1: Divide the normalized fingerprint image (N) into a block of size W x W

Step 2: Project the grey-level values of all the pixels (i, j) located inside each block along a direction orthogonal to the local ridge orientation

Step 3: Count the median number of pixels between consecutive minima points in the projection to obtain the ridge spacing $S(i, j)$

Step 4: Obtain the ridge frequency $F(i, j)$ for a block centered at pixel (i, j) such that:

$$F(i,j) = 1/S(i,j) \quad (2.9)$$

Step 5: Stop

}

GaborFiltering(image *O*, image *F*, image *N*)

{

Step 1: Obtain the enhanced image E using Gabor filter as follows

$$E(i,j) = \sum_{u=-wx/2}^{wx/2} \sum_{v=-wy/2}^{wy/2} G(u,v;O(i,j),F(i,j))N(i-u,j-v) \quad (2.10)$$

where O is the orientation image, F is the ridge frequency image, N is the normalized fingerprint image, and w_x and w_y are the width and height of the Gabor filter mask, respectively

Step 2: Stop

}

Binarization(Enhanced_image *E*)

{

Step 1: Divide the enhanced image (*E*) into a block of size *W x W* e.g 8 x 8 and set k to one i.e k = 1

Step 2: Calculate the average grey-level value for the block *k* i.e 8 x 8 such that

for i = 0 to N-1 where N is the size of the block *k*

for j = 0 to N-1

sum = sum + *E(i, j)* where *E(i, j)* is the gray-scale level value at pixel (*i, j*)

end j

end i

$T_k = 1/N^2$ * sum where T_k is the average grey-level value for block *k*

Step 3: Threshold the left-most part of block *k* by comparing T_k with each pixel (*i, j*) in *E(i, j)* such that:

If($E(i, j) < T_k$) then

- Set the pixel to zero (i.e black) where black is ridges

Else

- Set the pixel to one (i.e white) where white is valleys or furrows

Endif

Step 4: Threshold the right-most part of block *k* by comparing T_k with each pixel (*i, j*) in *E(i, j)* for that part using the same method in Step 3

Step 5: Increase *k* by 1 and repeat Step 2 to 4 until all the blocks have been threshold

Step 6: Output the fingerprint image as a binarized fingerprint image (*B*)

Step 7: Stop

}

Thinning(binarized_image *B*)

{

Step 1: Compute the block directional image such that:

For i = 0 to N-1 where N is the number of rows in the image

For j = 0 to M-1 where M is the number of columns in the image

- Find any black pixel in the image and assign this pixel as Ridge Meeting Point (RMP) which might be a point on the ridge

While (RMP = true)

- Find Ridge Continuity Points (RCP) on the ridge, the thinned points of the ridge.

- Check ridge spacing at each step, if ridge spacing changes rapidly, look for a bifurcation point.

- During this operation, remove the black pixels from original image,which is already processed by the algorithm.

if all pixels are processed on the ridge then

Assign RMP = false

else

Assign RMP = true

Endif

endwhile

end j

end i

Step 2: Output the thinned image

Step 3: Stop

}

Fingerprint Minutiae Feature Extraction

After the fingerprint image enhancement, the next stage is to extract the minutiae features of the enhanced fingerprint image. The algorithm for this extraction using Crossing Number method is described below:

MinutiaeExtraction(thinned_image *T*)

{

Step 1: Divide the thinned image *T* into windows of size 3 x 3 and set k = 1

Step 2: Center the k^{th} 3 x 3 window at point *P*

Step 3: Compute the crossing number CN_k for the window *k* such that

$$CN_k = 0.5 \sum_{i=1}^{8} |Pi - Pi + 1|, P9 = P1 \qquad (2.2)$$

If (CN_k = 1) then

- Assign the center point *P* in the window *k* as ridge ending

Elseif (CN_k = 3) then

- Assign the center point *P* in the window *k* as ridge bifurcation

Endif

Step 4: Increase *k* by 1 and repeat Step 2 to 3 until all the 3 x 3 windows have been scanned

Step 5: Output the new image

Step 6: Stop

}

Fingerprint Template Generation

This stage extracts the minutiae features of the fingerprint image and generates a template from these features. The algorithm for this is described below:

FingerprintTemplate(thinned_image *T*)

{

Step 1: Scan the thinned fingerprint image *T* and generate template *Tp* from the image such that

for i = 0 to N-1 where N is the number of rows,

for j = 0 to M-1 where M is the number of columns

- Start scanning the thinned fingerprint image *T* from the origin point line by line and find any black pixel in the thinned image

- A window of size of 3 x 3 is centered at the point

- Count the number of black pixels within the 3 x 3 window as *N*

if ($N = 2$) assign this point as ridge endings

if ($N = 3$) assign this point as ridge continuity

if ($N > 3$) assign this point as bifurcations

end j

end i

Step 2: Output the Template *Tp*

Step 3: Stop

}

Enrollment

This is the stage where the template of the user fingerprint will be saved into the database along with other user's details for subsequent authentication. The algorithm is shown below:

Enrollment(Template *Tp*, array biodata[ ], array otherinfo[ ])

{

Step 1: Connect to the database

Step 2: Create a new user record in a user table and save both the array element biodata[ ] and the user fingerprint template *Tp*

Step 3: Save the user's other info if any into the database

Step 4: Close the database connection

Step 5: Stop

}

Fingerprint Matching

At this stage, the user's fingerprint is captured and compare with the stored templates in the database. If the fingerprint corresponds to any of the enrolled fingerprint templates, access to the protected data or information will be granted, otherwise, the user will be denied access. The algorithm is described below:

FingerprintMatching(Template *T1*, Template *Tp*)

{

Step 1: Set $N = 1$, $S_{count} = 0$

Step 2: Obtain the minutiae sets *S1* and *S2* for each fingerprint template *T1* and Tp^N respectively where *T1* is the test template and Tp^N is the enrolled template at position *N* in the database.

Step 3: For each of the sets *S1* and *S2*, obtain the minutiae type *e1* and *e2*, coordinates *c1* and *c2* corresponding to the coordinate of the minutiae type, and the orientation angles *o1* and *o2* of the minutiae points for sets *S1* and *S2* $\in$ *T1* and Tp^N respectively.

Step 4: Compare if *S1* and *S2* are paired such that:

If (TYPE (*e1*) = TYPE (*e2*)) and (DIST (*c1*, *c2*) $\leq D_T$) and (ANGLE (*o1*, *o2*) $\leq A_T$) then

Increase S_{count} by 1

Goto Step 3 until no more sets to pair

Else

Goto Step 3 until no more sets to pair

Endif

Here, D_T and A_T are the maximum tolerance for translation and rotation respectively.

Step 5: Obtain the total number of sets in *T1* and call it *N1*

Step 6: Obtain the total number of sets in Tp^N and call it *N2*

Step 7: Compute the similarity measure *M* between *T1* and Tp^N such that:

$$M = \sqrt{Scount} * Scount/(N1 * N2) \qquad (2.3)$$

Step 8: Compare the source of the two fingerprint images such that:

If (*M* >= *T*) then (where *T* is a threshold)

- The two fingerprint images *T1* and Tp^N are from the same source
- Set Match = True
- Goto Label

Else

- The two fingerprint images are not from the same source
- Set Match = false
- Increase *N* by 1
- Goto Step 2 //this is necessary since identification method will be used

Endif

Exit

Label: Return Match

Step 9: Stop

}

4. Conclusion

This paper presents a review of the various stages involved in fingerprint identification using binarized fingerprint image. Algorithms for the different stages are presented. Experience has shown that there are several problems attributed to the use of identity cards, paper sheets, and knowledge based techniques in uniquely identifying individuals. Employing biometric for individual identification provides us with a batter approach for securing important data and information.

The naturalness in the use of fingerprint as a biometric trait makes it a reliable access control technique. The fact that individual no longer needs to memorize password, carry identity cards, and other documents for identification explain the ease of use of biometric system.

Acknowledgement

The authors acknowledge the comments of the reviewers for their suggestions and comments in improving the quality of the manuscripts.

References

Anil, K. J., Arun, A. R. & Patrick, F. (2008): *Handbook of Biometrics*. USA: Springer, 1–564.

Cappelli, R., Maio, D., Maltoni, D., Wayman, J. L., & Jain, A. K. (2006): Performance evaluation of fingerprint verification systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **28**(1), 3-18.

Chirillo, J. & Scott, B. (2007): *Implementing Biometric Security.* Indianapolis: John Wiley Publishing Inc., 1–432.

Erikson, M. (2001): *Biometrics Fingerprint based identity verification*. M.Sc. Thesis, Department of Computing Science, UMEA University.

Ghazali, B. S. (2005): Design and Development of an Automated Fingerprint Verification System. Retrieved February 3rd, 2013 from http://www.eprints.utm.my/4348/1/74021.pdf

Naser, Z. (2011): Minutiae-based Fingerprint Extraction and Recognition. Kuwait: Arab Open University. Croatia: InTech., Chapter 3, 55-79.

Raymond, T. (2003): *Fingerprint Image Enhancement and Minutiae Extraction*. M.Sc. Thesis, School of Computer Science and Software Engineering, University of Western Australia, 1 - 63.

Roli, B., Priti, S. and Punam, B. (2011): Minutiae Extraction from Fingerprint Images- a Review. *International Journal of Computer Science Issues*, **8**(5), 74-85.

Sharat, S. C. (2005): *Online Fingerprint Verification System*. M.Sc. Thesis, Department of Electrical Engineering, State University of New York, Buffalo.

Sozan, A. M. (2011): Fingerprint Identification Based on Skeleton Minutiae Extraction. ICGST AIML, 161-165. Retrieved 9th October, 2013 from http://www.icgst.com

Sunny, A. S. and Rudi, T. Y. (2012): Adaptable Fingerprint Minutiae Extraction Algorithm Based on Crossing Number Method for Hardware Implementation Using FPGADevice. *International Journal of Computer Science, Engineering and Information Technology*, **2**(3), 1-30.

SecuGen (2013): A state-of-the-art fingerprint identification solution that allows you to instantly identify employee, customers and partners. Retrieved 20th April, 2013 from http://360biometrics.com/faq/fingerprint_scanners.php.