An Accelerated Architecture Based on GPU and Multi-Processor Design for Fingerprint Recognition

Mossaad Ben Ayed
College of Science and Humanities at AlGhat, Al Majmaah University
Kingdom of Saudi Arabia

Sabeur Elkosantini
College of Engineering, King Saud University
Kingdom of Saudi Arabia

***Abstract*—Fingerprint recognition is widely used in security systems to recognize humans. In both industry and scientific literature, many fingerprint identification systems were developed using different techniques and approaches. Although the number of conducted research works in this field, developed systems suffer for some limitations partially those related the real time computation and fingerprint recognition. Accordingly, this paper proposes a reliable algorithm for fingerprint recognition based on the extraction and matching of Minutiae. In this paper, we present also an accelerated architecture based on GPU and multi-processor design in which the suggested fingerprint recognition algorithm is implemented.**

***Keywords*—Minutia; Fingerprint; Architecture design; recognition; Gabor filter; MPSOC**

I. INTRODUCTION

Individual identification presents a challenge for the modern society. In this context, biometric recognition presents the most popular method for identification. Particularly, fingerprint technique is the most used in industries for many reasons: cheap, secure and easy to deploy. Fingerprints are rich in details and contain a different form based on ridges. These forms define many characteristics point named "Minutia". Each individual has unique repartition of Minutia which is different than others. Consequently, fingerprint is always used in many systems as the identifier of humans. Minutia is defined as a local ridge characteristic. Fingerprint contains various types of Minutia, but usually two types of Minutiae are used: Termination and Bifurcation. Termination is defined as the end point of a ridge. Bifurcation is defined as the point where a ridge merges or splits into branch ridges [1].

A. Fingerprint recognition process

In the scientific literature, the process of fingerprint recognition is always divided to different phases including: the pre-processing, the extraction of the Minutiae and the matching (see Fig. 1).

The objective of the first phase, the pre-processing, is applied on gray-scale images essentially to improve and divide fingerprints ridges from the background texture. The first step is to apply a filter algorithm. This step is important to ameliorate the quality of the image and to decrease noise [2]. Then the binarization step transforms the image into binary and lets the separation between ridges and background easier. The last step in the pre-processing step is the skeletonization which is based on thinning algorithms. The aim of this step is to thin ridges to only 1 pixel wide. This preserves the essential information (Minutiae) with low size of storage. Also thinning algorithms reduces the data that represents Minutiae and make the treatment more effective and faster. In the scientific literature, there are many iterative methods for thinning including sequential [49] or parallel [3]. In this step, the most used window size is a 3X3 pixel. In this case the central black pixel has 8 neighbours that can be considered [3].

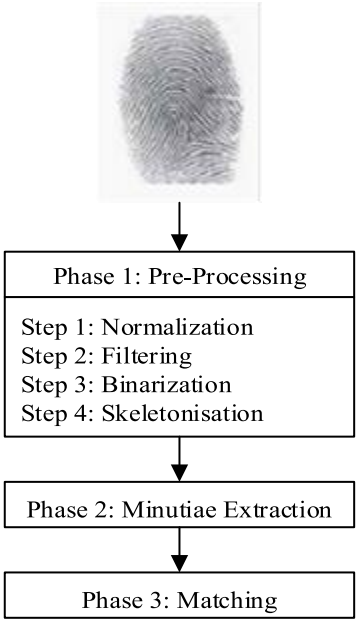


Fig. 1. Fingerprint recognition phases

The second phase is the extraction of the Minutiae. Many extraction methods were developed including those based on the nearest neighbour pixels around the central pixel [4]. Another method of extraction Minutia is presented in [5] which search the characteristics point Minutiae using thinned ridges. Other algorithms were based on classifier techniques [20]. [6, 7] tried to detect Minutiae using the ridge line without apply the thinning algorithms. In these works, different rules and ad-hoc methods are used to handle problems met on extraction. The extraction method proposed in [8] is based on Data-driven Error Correcting Output Coding (DECOC) classifier. This method presents a many advantages with other methods used. However, the outputs results depend not only on selected extraction algorithm but also on the pre-processing phase essentially binarization and skeletonization.

Matching phase aims to carry out fingerprint verification

and their generalization to find the factor of similarity between the input image and the corresponding database. This phase still suffers from the increasing of the time consumption, especially when the number of saved fingerprint is huge [42]. Although the number of the conducted research in the field of Minutiae matching for acceleration [48], many proposed architectures are not suitable for large database to maintain precision and the real time decision.

B. Problem statement

As mentioned bellow, the pre-processing and the extraction Minutiae are an important phases to improve not only the quality of the image but also the recognition result. Accordingly, many research works were conducted to develop techniques and algorithms. Unfortunately, in spite of the number of research works in this field, there is still a lack for systems allowing the real-time fingerprint recognition. In [50] [51], the authors propose matching algorithm based on fingerprint database. But these algorithms let essentially sharing data with database and not suitable for real time identification for large database.

In this context, the objective of this work is the development of a new accelerated architecture based on GPU and multi-processor design based on a new algorithm for fingerprint recognition based on the extraction and matching of Minutiae.

The remainder of this paper is organized as follows: Section 2 introduces the developed pre-processing algorithm. Section 3 presents the proposed Minutiae extraction based on DECOC classifier. Section 4 describes the Minutiae matching phase and presents the adopted method. Section 5 presents the implementation of the suggested system and the obtained results. The paper is concluded by a conclusion and suggests some future works.

II. Pre-Processing Phase

During fingerprint recognition, the image pre-processing represents an essential phase. The collected images from fingerprint sensor are challenged by the quality of fingerprinted person due to many conditions as skin condition, collection conditions, and the environment.

This phase can be divided into four steps: Normalisation, Filtering, binarization and skeletonization.

A. Normalization step:

Normalization step can be defined as a method to reduce the diversification degree of the grayscale image captured by the sensor. As known, the fingerprinted is composed by ridges and valleys that form the structure of texture information. The aim of the normalization step is to facilitate the pattern capture and the fingerprint frequency. But this step still suffers from confusing between ridges and valleys [24].

The normalized step is based on equation (1).

$$G(i,j) = \begin{cases} M_0 + ℵ \times VAR \times \left|I_{(i,j)} - M\right| & I_{(i,j)} > M \\ M_0 - ℵ \times VAR \times \left|I_{(i,j)} - M\right| & Others \end{cases} \quad (1)$$

B. Filtering step: Gabor

Filtering step is based on Gabor filter [13], [14], [21]. This filter can represent the local frequencies as mentioned in equation (2). It has both orientations: spatial domain and frequency domain. Developer can separate fingerprints from the background using Gabor filter algorithm based only on spatial domain. Indeed, the related works use only spatial domain because the Gabor filter take much time.

In this paper, spatial and frequency domain is used ensure the best filtering results [9]. This choice is adopted due to the proposed architecture that is based on GPU accelerator.

Gabor filter has the following equation:

$$h(x,y,\varphi,f) = e^{-\frac{1}{2}\left(\frac{x_\varphi^2}{\delta_x^2}+\frac{y_\varphi^2}{\delta_y^2}\right)}.\cos(2\pi f x_\varphi)$$

$$avec \begin{cases} x_\varphi = x\cos\varphi + y\sin\varphi \\ y_\varphi = -x\sin\varphi + y\cos\varphi \end{cases} \quad (2)$$

where

f : frequency of the sinusoidal of the plane wave

φ : orientation

δx,δy: space constants of the Gaussian envelope along x and y axes, respectively.

C. Binarization

Binarization step convert image from grayscale image to binary image. The most used method of binarization [15] is based on global threshold that consists in calculating a unique threshold for the total image.

The disadvantage of this method is shown when the fingerprint has different quality. In this case, binarization algorithm based on global threshold will eliminate many parts of the finger image (Figure 2.a).

This paper proposes a binarization method based on local threshold (Figure 2.b). This method is described as follows:

1) Divide the fingerprinted image into masks of 10 x 10 pixels.

2) Calculate the threshold of every mask. The last is determined using the equation (3).

$$Threshold = \frac{\sum M(i,j)}{nl * nc}$$

Where B(i, j) is the image

L * C is the number of pixels (3)

This method shows good results especially when fingerprints and background will be separated. This method is more adopted in the field of fingerprints because the partition of the pixel intensity is not homogeneous.

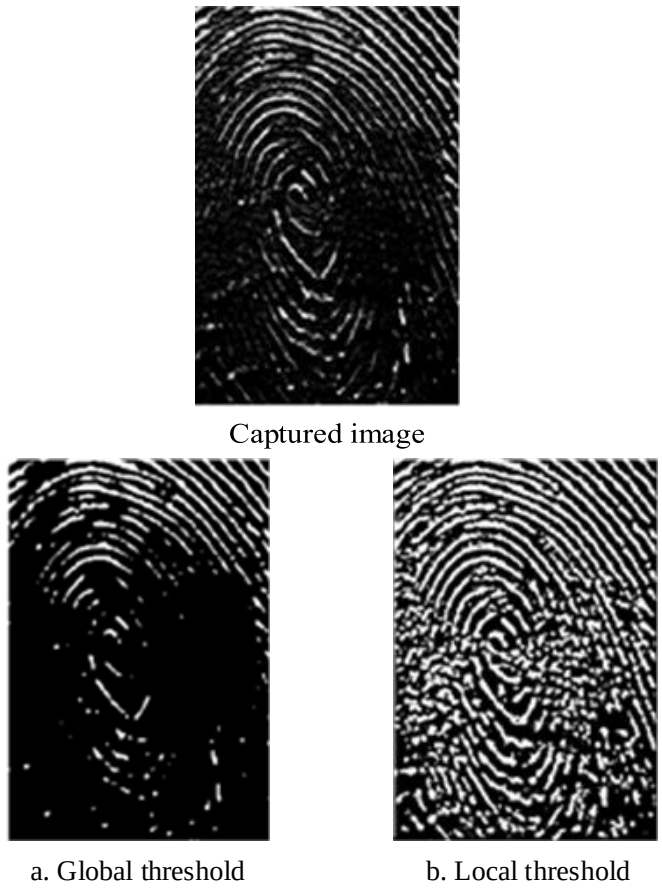


Fig. 2. Binarization methods

D. Skeletonization

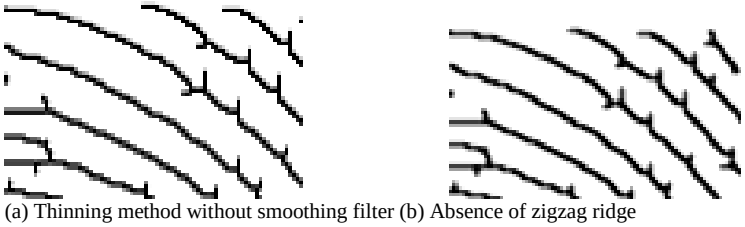
Skeletonization method is defined as a thinning algorithm that can be transforming the thickness of ridges into a single pixel [24].

The method used for skeletonization [16] is based on scanning the image by 3 x 3 blocks neighbourhood [17]. When this method is applied for the total of the image, the process costs too much time.

At first, this paper proposes a modified thinning algorithm like [17] to reduce the execution time of skeletonization step. For this reason, we consider that the process of thinning is applied only when the 3 x 3 bloc contains more than two black pixels. An experimental result shows that the call of thinning function is reduced on third. Figure 3.a shows the results of skeletonization process.

As observed by figure 3.a, the zigzag ridge presents the major problem of skeletonization. It can influence the detection of the Minutiae because the changing of one pixel can modify the kind of Minutiae.

As a second contribution in skeletonization, smoothing filter is applied to decrease the zigzag effect, see figure 3.b.



(a) Thinning method without smoothing filter (b) Absence of zigzag ridge

Fig. 3. Skeletonization method without (a) / with (b) Smoothing filter

III. MINUTIAE EXTRACTION PHASE

Minutia extraction presents the second phase. This phase aims to find a particular point or feature in fingerprinted image named as Minutia. The last can be defined as the intersection of ridges or the ending of ridges. Minutia can be classified as two parts: (1) Bifurcation Minutia (intersection of two ridges) (2) Termination Minutia (ending ridges).

A. Discussion

Minutia extraction is based essentially on classification stage that aims to detect features [25]. There are many related work that propose algorithm to identify the Minutiae grayscale. In [26], Minutia is determined using a mask sized 3 x 3 bloc. A Minutia is identified if the pixel has one or more than two neighbors. But when the quality of fingerprinted image is poor is extremely unreliable. There many work that lets Minutia extraction by using different classifier as SVM [36], Bayes [37], Neuron, Fuzzy, etc. Table 1 resumes the most important contributions.

TABLE I. COMPARISON BETWEEN DIFFERENT METHODS FOR MINUTIAE EXTRACTION

	FAR	**FRR**
HAO GUO method [32]	4.18%	9.93%
OMER SAEED method [33]	1,12%	Not indicated
Ying HAO method [34]	1%	2.5%
Jiong Zang method [35]	0.04%	1.31%
Ching-Tang and al. method [36]	0.5%	0%
Mossaad and al. method [27]	**0%**	**0.02%**

In this paper, the DECOC classifier is used for Minutia extraction based on comparison in [27]. This classifier is adopted in many fields such as manuscript recognition. For this reason it is applied for fingerprint recognition. In the next section, a brief definition for the adopted method is presented.

B. Error Correcting Output Coding (ECOC)

In this section, Error correcting output code algorithm will be described. The DECOC classifier is based on ECOC classifier. The last have been used in different fields as network communication and information theory. The main purpose of the use is the reliability of transmitting binary signals and the integrity information. The idea is to add the superfluous parity bits to an information word. The novel word result is named code word. The code word presents a binary code string. Then, the ECOC algorithm calculates the distances between two code words using Hamming distance, as shown in equation (4). The last is determined by the count of the different bits in the two patterns.

Hamming distance:

$$y = \arg_{\min(j)} H(w_j, w(x)),\ j = 1, \ldots, J \quad (4)$$

where w_j is the ideal code word assigned to group j.

H (w_j, w(x)) presents the main function that computes the Hamming distance between w_j and w(x).

Data-driven presents an extending version of ECOC. The aim is to provide new solutions to the problem of multi-class. In this fact, DECOC compared with ECOC will be applied to two multi-class pattern recognition problems [8].

C. Methodology: Data-driven ECOC

Data-driven ECOC (DECOC) chooses the code words utilizing specific information from the training data. For a K-class problem as pair wise coupling, a decomposed algorithm is proposed. Also a K*(K-1)/2 base learners are always needed.

This measure determines the learner that will be included in the ensemble. This measure is called confidence score. DECOC classifier is based essentially on two parameters: Separability criteria and confidence score [8].

- *Separability Criterion*

$$S(G) = \begin{cases} \dfrac{2}{|G|^2 - |G|} \displaystyle\sum_{0\, j \neq k, c_j c_k \in G} d(c_j, c_k) & |G| \neq 1 \text{ and } |G| \neq K-1 \end{cases} \quad (5)$$

- *Confidence Score*

$$C(f) = \begin{cases} \dfrac{S(G_{+/-}(f))}{S(G_+(f)) + S(G_-(f))}; |G_+| \neq 1 \text{ and } |G_+| \neq K-1 \end{cases} \quad (6)$$

As mentioned before, the use of DECOC classifier is to make decision about the kind of pixels blocks. This classifier can be defined into two parts as follow:

```
/***** Calculate inter-class distance *****/
int *Inter_class_distance(int TD[x][y])
{
        for (i=vli;i<vli+5;i++)
                for (j=vci;i<vci+5;i++)

        *Dist=Hamming(TD[i][j],TD[i+5][j]);
        return Dist;
}
/***** Calculate separability criteria *****/
int * Separability_criteria(int *Dist)
{
        for (i=1;i<len;i++)
                sum[i]+=Dist[i];
        for (i=1;i<len;i++)
                if (condition 1)
                SC[G1][i]=Separability_Factor * sum[i];
                if (condition 2)
                SC[G2][i]=Separability_Factor * sum[i];
                if (condition 3)
                SC[G3][i]=Separability_Factor * sum[i];
return SC;
}
```

Code 1: Inter-class distance and separablity criteria functions

- Part 1: Find the best base learners based on higher confidence score. At first, the interclass distance and separability criteria are calculated as mentioned in code (1). Then, the confidence score is calculated and applied to the sort algorithm as shown in code (2). The adopted base learner corresponds to the top of confidence score.

```
/***** Calculate confidence score *****/
int *confidence_score(int *SC)
{
        for (i=1;i<len;i++)
                CS[i]= SC[G1][i]/(SC[G2][i]+SC[G3][i]);
return CS;
}
/***** Sort confidence score *****/
int *sort_CS (int *CS)
{
        CS_Max=CS[sort(CS)];
        Pos_xy=*Pos(CS_Max);
return Pos_xy;
}
```

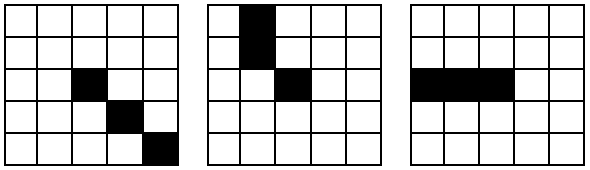
Code 2: Confidence score and sort functions

- Part 2: The decision is based in the lower hamming distance. We calculate the Hamming distance between the training data and the testing sample as defined in code (3).

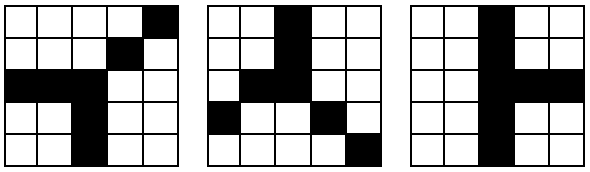
```
/***** Calculate Hamming Distance *****/
int *Hamming_distance(int *TD, int * TS)
{
for (i=vli;i<vli+5;i++)
        for (j=vci;i<vci+5;i++)
                Dist_TD_TS[i]=
Hamming(TD[i][j],TS[i][j]);
return Dist_TD_TS;
}
```

Code 3: Hamming distance function

In our case, the training data can be presented into three different pattern classes: (1) termination, (2) bifurcation and (3) non-Minutiae.



(a)Termination patterns



(b)Bifurcation patterns

Fig. 4. Sample of different patterns

The used size mask for image is the 5 × 5 pixels; it represents more information than the 3 × 3 mask and sufficient information than the 7×7 pixel mask. It is necessary that the size of the mask be an odd number to ensure the presence of central pixel.

To Apply the DECOC classifier, we have created 136 different patterns for two classes. 32 constructed patterns for termination class, 104 patterns for bifurcation class. Each pattern sizes 5 × 5 pixel. The third class, non-Minutia, is detected when an input pattern is not considered as termination class or as bifurcation class. Figure 4 presents some constructed patterns for the class Termination (a) and for the class Bifurcation (b).

IV. The Minutiae Matching Phase

Matching phase can be presented as a mechanism that is responsible to make decision based on likeliness parameter between two fingerprints. There are different methods in literature that lets Minutiae matching based on types of features [29][30][31] or on correlations of image [28].

The Minutiae matching phase aims (1) to make a reliable decision and (2) to ensure the real time response.

The first is faced with the problems of variations as the displacement of fingers, the rotation of fingers, the nonlinear distortion, the pressure of fingers when touch the sensor, the skin condition and feature extraction errors, etc [1], present a major problem.

Minutiae matching are classified in two families [28]:

- Global Minutiae matching; it aims to align Minutiae between two compared fingerprinted based on two directions and the angle. Works based on global Minutiae still suffer from on-precision and false identification [28].
- Local Minutiae matching; it aims to compare two fingerprints according to the relationships between proximity Minutiae. Works based on local Minutiae matching is more adopted than global Minutiae matching [28].

In the literature, there mainly three proposed method for matching phase. The first represents the classical matching algorithm [53]. In this case each Minutia is identified by its neighborhood Minutiae and the comparison is made by pairs. When a two pairs is detected as similar, the rest of comparison is made by coordinates and angles.

The second Minutiae matching algorithm [54] considers the topology of Minutia given by fixed radius and the comparison is ensured by the similarity of local topology.

The third Minutiae matching algorithm [55] gather the methods in [53] and [54]. Each Minutia is described using its position and the relative position of neighboring Minutiae.

To sum up, the last method is more precise but its time consumption is important.

This paper proposes a modified algorithm based in [55]. The proposed matching algorithm uses the location coordinates to extract new information/ relationship between Minutiae without be tied with fixed reference, as shown in figure 5. In our case, Minutia is presented by equation (7).

$$M = (x, y, \theta) \quad (7)$$

Where (x,y) is the coordinate of Minutia

ɵ is the orientation angles

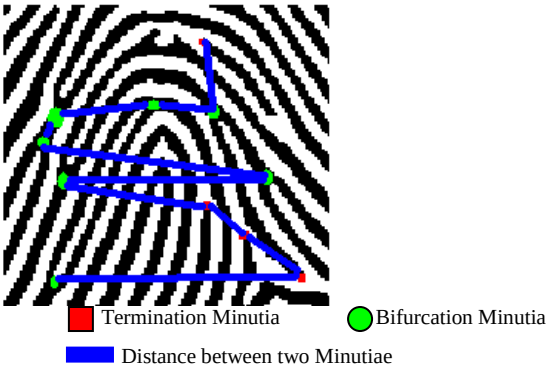


Fig. 5. The matching method

The matching method is based on three parameters:

- The Normalized Euclidean Distance between every two Minutiae is calculated according to equation (8),

$$Distance_{(M1,M2)} = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \quad (8)$$

where M1 and M2 two Minutia and (x,y) presents the coordinate of each point.

- Direction between two successive Minutiae, see figure 6, using equation (9), (10),

$$Direction_{(M1,M2)} = \frac{y_2 - y_1}{x_2 - x_1} \quad (9)$$

$$\theta = |\arctan(\text{direction1})| + |\arctan(\text{direction2})| \quad (10)$$

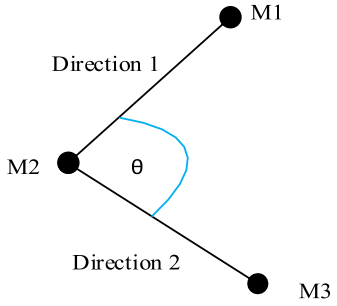


Fig. 6. Anlge between three Minutiae

- Types of successive Minutiae as mentioned in equation (11),

$$Type_{(M1,M2)} = \begin{cases} 11 & Bifurcation - Bifurcation \\ 10 & Bifurcation - Termination \\ 01 & Termination - Bifurcation \\ 00 & Termination - Termination \end{cases}$$

At conclusion, every fingerprint is associated with a unique signature that is composed by distance, type and direction between two Minutiae, see equation (12).

$$M = (Distance_{MM_{-1}}, Type_{MM_{-1}}, \theta,) \quad (12)$$

The matching algorithm process as follow:

1) Compare between Input Distance and the all stored distance in database. If the distance is lower than ε=0.01, we proceed to next step.

2) Compare between Input Type and the adequate stored type in database. If the comparison is equal we proceed to next step.

3) Compare between Input direction and the adequate stored direction in database. If the direction is lower than ε=0.05 the Minutia is accepted and it be considered a true Minutia.

The second aims of the Minutia matching phase in this paper, as mentioned above, is to ensure the real time execution. Many system based on identification is challenged by the real time decision and there are many related works in the literature that deals about it [42], [43], [44]. The real time execution is ensured by the identification for distributed database or by the time of identification.

In [50], the authors propose Minutiae matching phase based on client-server system when the fingerprint database is presented in many servers. But this kind of system doesn't support the parallel execution. That's why client-server system is not suitable for very large database.

In [51] and [52], the authors propose Minutiae matching phase based on agent-based systems. In fact this kind of system is adopted for the heterogeneous hardware architecture. The objective is to ensure load balancing between shared machines. Unfortunately, as mentioned in [51], the execution time is still important to guarantee the real time decision for large database.

The cited research works goals to provide identification for distributed databases, but this paper focuses to decrease the time of identification for large database. The state of arts is introduced in the next section.

This paper proposes, in the next section, an architecture based on GPU and multi-processor design for fingerprint recognition to accelerate the identification time.

V. Experiment Results

The implementation of fingerprint recognition represents a challenge especially for system that contains a huge number of fingerprints saved in database. Nowadays, the progress in term of technology became an advantage for recognition system to accelerate the consuming time. Fingerprint recognition involves many multiplications, evolution, rotation, and floating-point operations, which greatly slow the processing speed [24]. The implementation of fingerprint recognition system using hardware accelerator is widely used in literature [22] [23]. These implementations are much linked with the speed of the used processor and the time execution of the whole system is limited. In the next section, a brief resume for different hardware architectures is described.

- Hardware board; there are many work that implement the whole recognition algorithm into board based on microcontroller as [38]. But this solution suffers from the increasing of execution time above all when the database exceeds about one thousand.

- Embedded systems; this solution is adopted for intelligent sensors [39]. In this case the sensor encloses the whole architecture to make decision about recognition. Embedded systems are used for a limited number of users.
- Hardware acceleration fingerprint; this solution is performed to speed up filtering and matching step by using: FPGA-based [40], GPU-based [41] [43], parallel architecture [42].

In this paper, we propose to combine into hardware accelerator and multi-processor architecture. The next section presents the multi-processor architecture.

A. Multi-Processor architecture

The image of fingerprint is received by sensor with size of 200x300 pixels and divided into mask sized by 10x10 pixels. The proposed design is composed by SRAM memory, control unit, 4 processors elements (cores) and 4 hardware accelerators based on Gabor filter, see figure 7. The proposed architecture belongs to SIMD/MPSOC field [44]. The control unit represents an essential component in design. The last aims to handle all processes:

- Arbitrating the access processors units to/from memory;
- Handling and commanding all the processor unit;

At first, the image is captured with the fingerprint sensor and saved into principal memory. The proposed algorithm for fingerprint recognition is divided into software modules (skeletonization and Minutiae extraction) and hardware components (Filtering, Binarization and Matching), see the next section. In this case, the skeletonization and the extraction Minutiae code are saved, also, into principal memory. The control unit assigns the right process for processors elements. The proposed MPSOC architecture is composed by Microblaze soft-core processors interfaced with shared memory using the AXI4 bus [45]. All processors elements, on-chip memories, and the AXI4 bus are clocked at 100 MHz. Off-chip memory is clocked at 200 MHz and the AXI-lite bus is clocked at 50 MHz.

B. HW/SW implementation

This section presents the implementation of the recognition algorithm using Co-design approach. The aim of this section is to describe a hardware accelerator based on Gabor filter.

Step 1: Partitioning

Partitioning phase is based on execution time of different module of fingerprint recognition as shown in figure 8. The rule of partition suppose that the module where spend more time will be considered as hardware component and the module where spend less time will be considered as software parts [18]. Then, we find the result shown in figure 8 based on the native execution of the fingerprint recognition on a 6 Giga Byte memory, 2.2 GHz frequency of the processor Intel Core i3 with Windows 7 as operating system. We notice that the time execution of the Minutia extraction module is the minimum and we can divide our architecture into:

- Hardware components: Binarization, Filter and Matching.
- Software applications: Minutia extraction and Skeletonization.

Step 2: Gabor filter design

Gabor filter can be divided into three essential parts: Control Unit (CU), Arithmetic Unit (AU) and Memory [11]. The proposed architecture is based on ‘CONV’ signal that precede the convolution operation of the filter. When the ‘CONV’ signal is selected, the convolution process starts. When it is not selected, the filter proceeds in reception phase. In this phase the filter receives data that is corresponding on image input from DATAIN. The storage will be in the memory, see figure 9.

Simulation shows that the step of filtering consumes more time in execution than other steps. The idea is to implement a digital Gabor filter as a hardware accelerator. The contribution is to describe the Gabor phase that was never exploited [9] in the literature. There are three essential parameters that must be considered under design: (1) accelerate the speed of execution (2) minimize the Silicium area and (3) decrease the power consumption. Firstly, when the convolution signal is not selected the input data receives pixels and stores them in the memory [11]. When the convolution signal is selected the convolution process starts and follows steps mention in figure 10. The output image that is filtered is given after nine convolution operation.

Figure 11 presents the overview of the top level filter. Gabor filter is composed by six inputs and one output. Data IN receives initial data before filtering. During the writing phase in memory, P-X and P-Y will be stored in specific location. CLK input generates signal every 40ns period. RST input resets the execution of the filter. Arithmetic Unit presents the core of the digital Gabor filter, see figure 11. This unit store the coefficient of Gabor [4]. AU is composed by necessary 2 parts: the ROM and the MAC.

- The main advantage of the ROM as a memory is to save the coefficient.
- The MAC is the association of nine multipliers and eight adders. The multipliers are designed in parallel to accelerate the convolution process. The adders are designed in sequence to make the sum of the nine multipliers.

The ‘CONV’ signal ensures that the convolution process has made without any mismatch and with the correct data. The ‘CLK’ signal and the ‘CONV’ signal are connected to the memory and the MAC. When the CU receives ‘READY’ and ‘SET’, the convolution process finishes. The CU sends an activation signal to the ‘CONV’ to activate the convolution process. The convolution function takes 9 complete processes. Before the multiplier operation, the ‘CON’ signal data will be stored in a buffer. The idea is to be sure that the data sent from the memory to the convolution process is correct. The design is based on pipeline architecture. The duration of execution contains 222 clock cycles with a time period of 40ns.

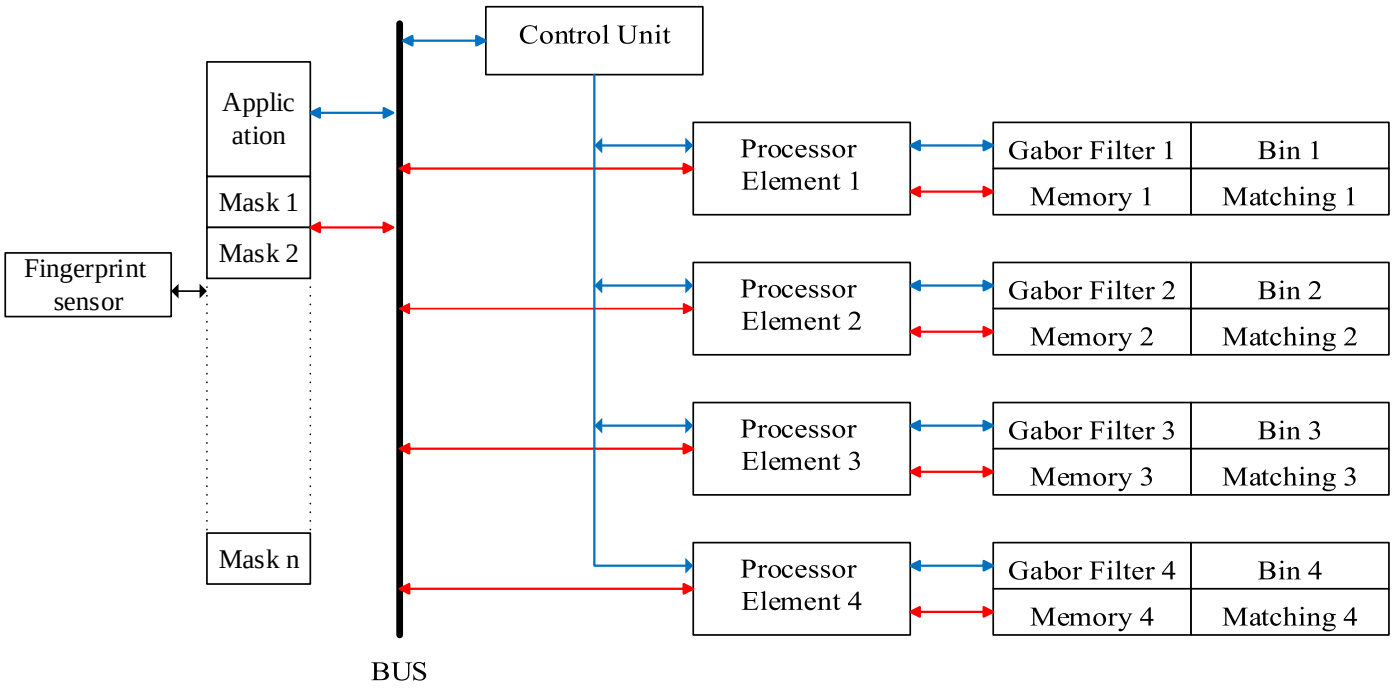


Fig. 7. Proposed MPSOC architecture

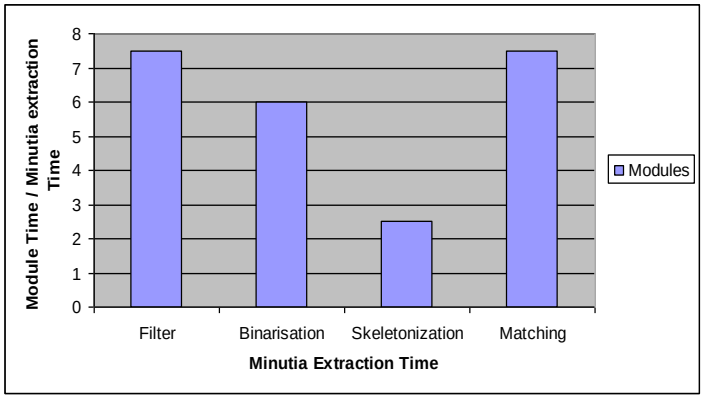


Fig. 8. Time execution of each modules as a function of Minutia extraction time

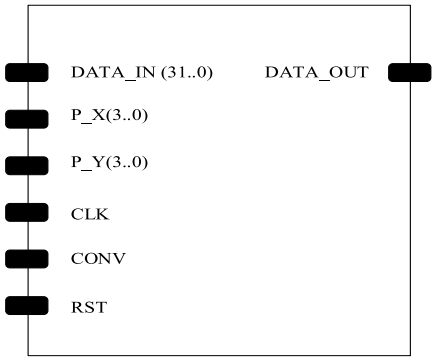


Fig. 9. Top level of Gabor filter

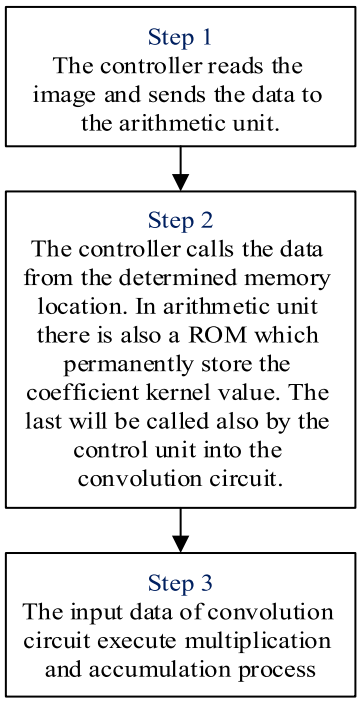


Fig. 10. Process Steps

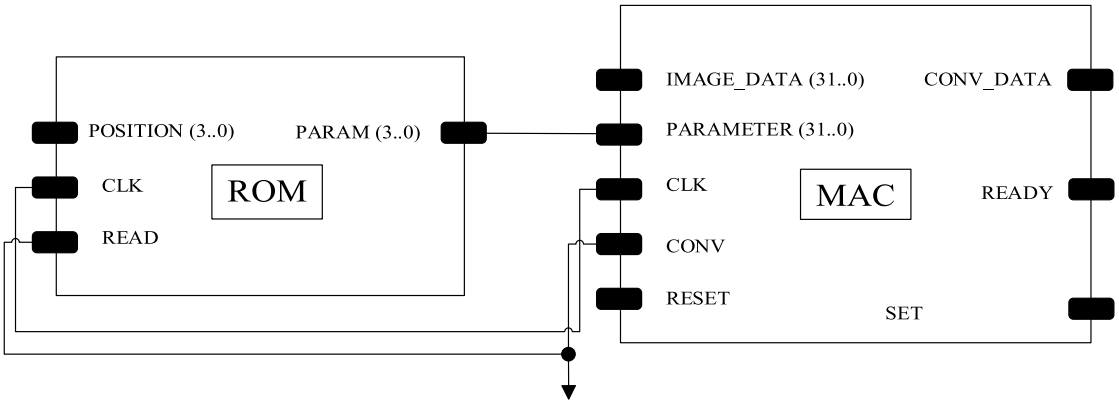


Fig. 11. Arithmetic unit

The control logic unit function presents the main controller in the filter. His role is to attribute process for other blocks. It plays also the role of middle between memory and AU.

The CU generates the address location if the 'START' signal is selected. The CU is composed by two blocks: counter for the coefficient and the counter decoder. The CU is composed by two blocks: counter for the coefficient and counter for the decoder.

The design of the counter gives the relationship between the coefficient and memory address.

The main role of the memory is to save data image in specific location. The address for X-direction is complete from two parts: CU or DATAIN.

C. Results and discussion

In this section, the results obtained from experiments are presented. Discussion section can be divided into two parts: GPU (Gabor filter, binarization and matching) experiments, multi-processor design experiments.

GPU (Gabor filter, binarization and matching) experiments;

The outputs results for the top level filter are the real convoluted data and the expected result. The difference between outputs results represents the error of convolution function. This error is only about 0.001%. The proposed design of Gabor filter based on amplitude and phase both shows a good performance. In addition, the execution time is speedy, it takes 222 cycles only to obtain filtered image. In term of silicium surface, experiments show also that the number of slices has been decreased from 5759 to 1625. This result is performed because in our proposed design we have reduce the number of multipliers and adders compare of the traditional architecture of Gabor filter. Our proposed design makes different optimization in memory functionality and the

controller unit.

Multi-processor design experiments;

DB14 represents a database that contains 10.000 fingerprints as templates. By a single identification, the recognition process computes 100 identifications using 100 different and randomly selected fingerprints from each database as the input [43]. The next table presents the results using the proposed algorithm and design comparing with others works. It is necessary to mention that all compared works use the same database and the same conditions [43].

In table 2, the proposed design ensures the best execution time for fingerprint recognition. Our proposition shows that the speed-up factors range is about 44.63 times in comparison with CPU execution in [43]. The result shows also that the speed-up factors range is about 2.84 times in comparison with GPU execution in [43], 1.84 times in comparison with Jiang design [46], and 1.43 times in comparison with Chen design [47], see figure 12.

TABLE II. COMPARISON BETWEEN THE PROPOSED DESIGN AND RELATED WORKS BASED ON TIME EXECUTION

	Execution Time (s)	***Speed-up in comparison with CPU [43]***
Execution based on CPU [43]	10.802	
Execution based on GPU [43]	0.688	15.69
Jiang Design [46]	0.446	24.21
Chen Design [47]	0.347	31.12
Our proposition	0.242	44.63

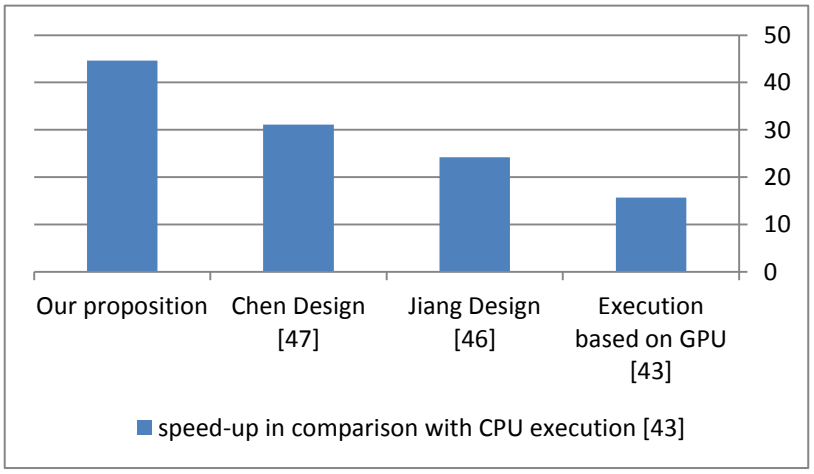


Fig. 12. speed-up factors range

VI. CONCLUSION

In this paper, an algorithm for fingerprints recognition is proposed. The last is based on comparison made between different works in literature. As a first contribution, a Gabor filter design has been proposed as hardware accelerator to decrease the execution time. The estimated results are reached. The experiment results between the outputs results of Gabor filter algorithm and the design is too near. This proves that the proposed blocks are well chosen.

As a second contribution, a whole hardware design based on multi-processor architecture is described. The proposed design combines with 4-cores and GPU (Gabor filter, binarization and matching) for hardware implementation and software modules: skeletonization and extraction Minutiae.

Compared with other works cited in previous section, our design achieves the best execution time. This advantage is showed mainly for ATM system that has a huge number of saved fingerprints.

ACKNOWLEDGMENT

The author would like to thank Deanship of Scientific Research at Majmaah University for funding this work.

REFERENCES

[1] d. Maltoni, d. Maio, a.k. jain, and s. Prabhakar, “handbook of fingerprint recognition”. Springer; 1st edition, 2003, isbn 0- 387-95431-7.

[2] b.g. sherlock, d.m. monro, and k. Millard, “fingerprint enhancement by directional fourier filtering,” vision, image and signal processing, ieee proceedings-, vol. 141, no. 2, 1994, pp. 87–94.

[3] p.m. patil,; s.r. suralkar,; f.b. sheikh, “rotation invariant thinning algorithm to detect ridge bifurcations for fingerprint identification”, 17th ieee international conference ictai 05 on volume, issue, 16-16 nov. 2005.

[4] n. Ratha, s. Chen, and a.k. jain, “adaptive flow orientation based feature extraction in fingerprint images,” elsevier pattern recognition issn 0031-3203 coden ptnra8, vol. 28, no. 11, 1995, pp. 1657–1672.

[5] a.farina, k. Vajna, and a.leone, “fingerprint minutiae extraction from skeletonised binary images,” pattern recognition, vol. 32, no. 5, 1999, pp. 877–889.

[6] d. Maio and d. Maltoni, “direct gray-scale minutiae detection in fingerprints,” ieee transactions on pattern analysis & machine intelligence, vol. 19, no. 1, 1997, pp. 27–41.

[7] x. Jiang, w.y. yau, and w. Ser, “detecting the fingerprint minutiae by adaptive tracing the gray-level ridge”, pattern recognition, vol. 34, no. 5, 2001, pp. 999–1013.

[8] m. Ben ayed, f. Bouchhima and m. Abid, “automated fingerprint recognition using the decoc classifier", international journal of computer information systems and industrial management applications. Issn 2150-7988 volume 4 (2012) pp. 546-553

[9] f.bellakhdhar , m.bousselmi and m. Abid, “face identification using the magnitude and the phase of gabor wavelets and pca”, 2012

[10] m. Liwen, y. Liang, q. Pan, h. Zhang “a gabor filter based fingerprint enhancement algorithm in wavelet domain”, proceeding of iscit 2005.

[11] p. H. W. L. Ocean y. H. Cheung, eric k.c. tsang, "implementing of gabor-type filters on field programmable gate arrays," 2005.

[12] b. Heisele, p. Ho, "face recognition: component-based versus global approaches". Compter visionand image understanding,2008.

[13] w. Kong, d. Zhang, w. Li, “palmprint feature extraction using 2-d gabor filters”, the journal of the pattern recognition society, elsevier, 2003, p 2339-2347

[14] m. Abadi, m. Khoudeir, s. Marchand, “gabor filter-based texture features to archaeological ceramic materials characterization”, image and signal processing, springer, 2012, volume 7340, pp 333-342

[15] s. Shaikh, a. Maiti, n. Chaki, “ a new image binarization method using iterative partitioning”, machine vision and applications, 2013, volume 24, pp 337-350.

[16] j. Teubner, l. Woods, c. Nie, “skeleton automata for fpgas: reconfiguring without reconstructing”, proceedings of the 2012 acm sigmod international conference on management of data, 2012, pp 229-240.

[17] a. Beristain, m. Grana, “a stable skeletonization for tabletop gesture recognition”, international conference on computational science and its applications, 2010, pp 610-621.

[18] l. Luo, h. He, q. Dou, w. Xu, “hardware/software partitioning for heterogeneous multicore soc using genetic algorithm” international conference on intelligent system design and engineering application, 2012, pp 1267-1270.

[19] j. Silva, k. Costa, v. Roda, “the c compiler generating a source file in vhdl for a dynamic dataflow machine”, international conference on mathematical methods and computational in electrical engineering, 2009, pp 33-36.

[20] pradeepta k. Sarangi, p. Ahmed, kiran k. Ravulakollu,” naïve bayes classifier with lu factorization for recognition of handwritten odia numerals”, international journal of science and technology, volume 7, issue 1, january 2014.

[21] m. Kocevar, s. Klampfer, a. Chowdhury, z. Kacic, “low-quality fingerprint image enhancement on the basis of oriented diffusion and ridge compensation”, international journal elektronika ir elektrotechnika, volume 20, issue 8 , 2014.

[22] m. Gok, s. Gorgunoglu, i. Muharrem orak, “fingerprint pre-processing on arm and dsp platforms”, international journal elektronika ir elektrotechnika, volume 20, issue 6, 2014.

[23] f. Yucel, o. Oral, n. Caglayan, m. Tecimen, s. Kocak, e. Yuce, “design and implementation of a personal computer authorization system using color detection”, volume 15 , issue 9 , 2011.

[24] j. Wang, l. Wu, y. Liu, " nios ii processor-based fingerprint identification system", nios ii embedded processor design contest—outstanding designs 2007.

[25] a t gowthami, h r mamatha, "fingerprint recognition using zone based linear binary patterns", procedia computer science, v.58, 552-557, 2015.

[26] s. Prabhakar, anil k. Jain and sharath pankanti. Learning fingerprint minutiae location and type.pattern recognition society. Published by elsevier science ltd,2003 .

[27] m. Ben ayed, f. Bouchhima, m. Abid, “a novel application of the classifier decoc based on fingerprint identification”, interactive multimodal pattern recognition in embedded systems impress 2010 workshop on database and expert systems applications dexa 2010.

[28] d. Peralta, m. Galar, i. Triguero, daniel paternain, salvador garcia, edurne barrenechea, josé m. Benitez, humberto bustince, francisco herrera, "a survey on fingerprint minutiae-based local matching for verification and identification: taxonomy and experimental evaluation", information sciences journal, p 67-87, 2015

[29] a.k. jain, j. Feng, k. Nandakumar, fingerprint matching, ieee comput. 43 (2010) 36–44.

[30] d. Maltoni, d. Maio, a.k. jain, s. Prabhakar, handbook of fingerprint recognition, second ed., springer publishing company, incorporated, 2009.

[31] n. Yager, a. Amin, fingerprint verification based on minutiae features: a review, pattern anal. Appl. 7 (2004) 94–113.

[32] h. Guo “a hidden markov model fingerprint matching approach”, proceedings of the fourth international conference on machine learning and cybernetics, guangzhou, ieee print isbn: 0-7803-9091-1 , 5055 - 5059 vol. 8 , august 2005

[33] o. Saeed, a. Bin mansoor, and m. Asif afzal butt “a novel contourlet based online fingerprint identification”, bioid multicomm2009, lncs 5707, pp. 308–317, 2009. Springer-verlag berlin heidelberg 2009.

[34] y. Hao, t. Tan, y. Wang “an effecitve algorithm for fingerprint matching” national lab of pattern recognition, cas, institute of automation, beijing, p. R. China.

[35] j. Zang, j. Yuan, f. Shi, s. Du “a fingerprint matching algorithm of minutia based on local characteristic”, isbn 978-0-7695-3304-9/08, ieee 2008.

[36] ch.tang hsieh and ch. Hu, "fingerprint recognition by multi-objective optimization pso hybrid with svm", journal of applied research and technology, volume 12, issue 6, december 2014, pages 1014–1024

[37] j. Hong, j. Min, u. Cho, s. Cho," fingerprint classification using one-vs-all support vector machines dynamically ordered with naı¨ve bayes classifiers ", pattern recognition, volume 41, issue 2, february 2008, pages 662-671

[38] m.a. murillo-escobar, c. Cruz-hernandez, f. Abundiz-pérez, r.m. lopez-gutiérrez, "a robust embedded biometric authentication system based on fingerprint and chaotic encryption", expert systems with applications, volume 42 (2015) 8198–8211

[39] s. Bayram, h.t. sencar, n. Memon, efficient sensor fingerprint matching through fingerprint binarization, ieee trans. Inform. Forensic. Secur. 7 (2012) 1404–1413.

[40] r.m. jiang, d. Crookes, fpga-based minutia matching for biometric fingerprint image database retrieval, j. Real-time image process. 3 (2008) 177–182.

[41] p.d. gutiérrez, m. Lastra, f. Herrera, j.m. benitez, "a high performance fingerprint matching system for large databases based on gpu", ieee trans. Inform. Forensic. Secur. 9 (2014) 62–71.

[42] d. Peralta, i. Triguero, r. Sanchez-reillo, f. Herrera, j.m. benitez, "fast fingerprint identification for large databases", pattern recogn. 47 (2014) 588–602.

[43] m. Lastra, j. Carabaño, p. D. Gutiérrez, j. M. Benítez, f. Herrera," fast fingerprint identification using gpus ", information sciences, volume 301, 20 april 2015, pages 195-214

[44] d. Watson, a. Ahmadinia, " memory customisations for image processing applications targeting mpsocs ", integration, the vlsi journal, volume 51, september 2015, pages 72-80

[45] s. Saadi, m. Touiza, f. Kharfi, a. Guessoum, " dyadic wavelet for image coding implementation on a xilinx microblaze processor: application to neutron radiography ", applied radiation and isotopes, volume 82, december 2013, pages 200-210

[46] x. Jiang,w.yau," fingerprint minutiae matching based on the local and global structures", proceedings of the15 th international conference on pattern recognition,vol.2,ieee,2000,pp.1038–1041.

[47] k. Cao,x.yang,x.chen,y.zang,j.liang,j.tian, "a novel ant colony optimization algorithm for large-distorted fingerprint matching", pattern recognition45(2012)151–161.

[48] r. Cappelli, m.ferrara, d.maltoni, "minutia cylinder-code: a new representation and matching technique for fingerprint recognition", ieee transactions on pattern analysis and machine intelligence 32(2010)2128–2141.

[49] nemeth, g., k. Palagyi., topology preserving parallel thinning algorithm. International journal of imaging system and technology, 2011: 21: 37-44.

[50] r.f.miron, t.s.letia ,m.hulea, "two server topologies for a distributed fingerprint-based recognition system" ,15th international conference on system theory, control and computing, 2011, pp.1–6.

[51] k. Beghdad bey, z. Guessoum, a.mokhtari, f.benhammadi, "agent based approach for distribution of fingerprint matching in a meta computing environment", proceedings of the 8th international conference on new technologies in distributed systems, 2008,pp.1–7.

[52] k. Nagaty, e.hattab, "an approach to a fingerprints multi-agent parallel matching system", ieee international conference on systems, man and cybernetics, volume5, 2004 ,pp.4750–4756.

[53] x. Jiang, w.yau , "fingerprint minutiae matching based on the local and global structures", proceedings of the 15th international conference on pattern recognition, vol.2, ieee, 2000, pp.1038–1041.

[54] x. Chen, j.tian, x.yang, "a new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure", ieee transactions on image processing 15(2006)767–776.

[55] r. Cappelli, m.ferrara, d.maltoni, "minutia cylinder-code: a new representation and matching technique for fingerprint recognition", ieee transactions on pattern analysis and machine intelligence (2010)2128–2141.