Improved Personal Identification Method Based on Partial Fingerprints

Radu Miron*, Tiberiu Leţia*

**Technical University of Cluj-Napoca, 28 Memorandumului St., Romania*
(e-mails: radu.miron@aut.utcluj.ro, tiberiu.letia@aut.utcluj.ro)

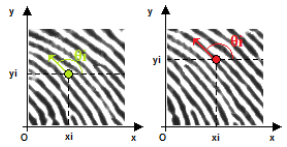
Abstract: The most common fingerprint recognition systems are those that are based on minutiae matching. Latent fingerprints lifted from different objects are usually noisy and broken, resulting in small and partial usable areas. Minutiae based techniques are widely used because of their temporal performances, but they do not perform so well on low quality images and in the case of partial fingerprint they might not be used at all. Therefore, when comparing partial input fingerprints with pre-stored templates, a different approach is needed. This paper proposes a fuzzy logic algorithm based on correlating a minutiae set and the regions between ridges for matching partial fingerprints.

Keywords: image processing, pattern recognition, identification algorithms, fuzzy logic, inference.

1. INTRODUCTION

Biometric recognition refers to both physiological and behavioural characteristics used to automatically recognize individuals. A number of different biometric identifiers (such as: iris, hand geometry, face, veins, voice, retina, handwriting, fingerprints etc) are used in various recognition applications. According to International Biometric Group (2009) the fingerprint based technology is the undisputed biometric leader, considering its market share of almost 67% in 2009. The same report forecasts that the annual biometric industry revenues will almost triple by 2014.

Fingerprint ridges begin to develop during the third to fourth month of fetal development. They are fully developed by the seventh month and the probability of two fingerprints being alike is 1 in 1.9 × 1015 (Leung et al. (1991)).



a) b)

Fig. 1. Minutiae types: a) termination, b) bifurcation.

Ridges and furrows are usually parallel, but they can also come to an end or split, thus creating the two most widely used types of minutiae (small details): *terminations* and *bifurcations*, respectively. There are other types of minutiae such as: pores, short ridges (dots or islands), cores, deltas etc. Although these other types of minutiae can be considered, the FBI minutiae-coordinate model uses only terminations and bifurcations (Wegstein (1982)). Usually a minutia is defined by the triplet $\{x_i, y_i, \theta_i\}$, where x_i and y_i are the minutia coordinates and θ_i is the angle between the tangent to the ridge line at the minutia location and the horizontal axis (refer to Fig. 1).

Besides minutiae, the other major fingerprint feature class is represented by the *singularities*, which are distinctive patterns that the ridges form in a fingerprint. Developed by Sir Edward Henry in the late 1800s, the *Henry Classification System* (Henry (1900)) is still used for fingerprint classification in order to simplify the search and retrieval process in the recognition system's data base. The main fingerprint typologies are: arch, loop and whorl. These typologies are further divided into subclasses (e.g. right loop and left loop, tented arch and plain arch).

The major steps involved in an automatic fingerprint recognition application are: the fingerprint acquisition, the fingerprint image pre-processing, the feature extraction, the fingerprint classification and the fingerprint matching.

There are many methods and technologies for each step involved in fingerprint recognition. The fingerprint matching is the process of comparing an input fingerprint to pre-processed ones (or templates) stored in a data base.

According to Maltoni et al. (2005), fingerprint matching can be grouped into three major classes: (i) correlation-based matching, (ii) minutiae-based matching and (iii) ridge feature-based matching.

High quality matching of complete fingerprints can be performed by many reasonable algorithms. Matching of poor quality or partial samples is more difficult. Fingerprint images can be affected by: high displacement and/or rotation, non-linear distortion (caused by representing a 3D shape in a 2D image), different pressure, skin condition and feature extraction errors (Maltoni et al. (2003)).

Minutiae based recognition techniques are the most often used methods in fingerprint recognition commercial applications because of their temporal performance, but they don't perform very well on low quality inputs (Marana (2005)). The loss of singular points (core and delta) is making singularity based recognition and indexing techniques impossible (Le and Bui (2009)).

The miniaturization of fingerprint sensors and their sensing areas raises a problem regarding the individual's fingerprint identification because of the fact that in most civilian

applications (PDA's, cell phones, laptops etc.) only a partial fingerprint can be captured. Another example of partial fingerprint acquisition is the lifted latent prints from a crime scene. In this case, the fingerprints are usually noisy and broken, thus reducing the usable areas (Jea, T. (2005)).

This paper proposes to differentiate between the matching approaches depending on the quality of the input fingerprint images; it is considered that the high quality of the pre-stored templates was ensured during the enrolment process. An input image is considered to have a fair quality if at least one singularity (loop, whorl or delta) can be extracted. In this case, a classical minutia based algorithm proposed by Zhang W. (2002) is used. The novel fuzzy logic based algorithm for partial fingerprint matching, proposed by this paper, combines correlation-based and minutiae-based techniques and it is employed if no singularity is detected in the fingerprint's image.

2. SYSTEM'S SOFTWARE ARCHITECTURE

The proposed algorithm can be integrated in a automated recognition system. The main steps involved in the partial fingerprint recognition algorithm together with the system that integrates it are presented in the logic diagram from Fig. 2.

The first step of the algorithm denoted by the block S1 is the fingerprint acquisition. The input acquisition can be performed either online, or a set of latent prints can also be given to the system. The application also allows the enrolment of individuals if the proposed fingerprint's quality is fair. The enrolment process is represented by the S15, ... , S18 blocks.

The proposed partial fingerprint matching algorithm is composed of the S3, ... , S14 blocks and it is described in the next chapters of this paper.

3. FINGERPRINT IMAGE PROCESSING

Before the proposed matching algorithm can be applied, some of the fingerprint's features must be extracted. The main steps involved in the fingerprint's image processing are presented in Fig. 3.

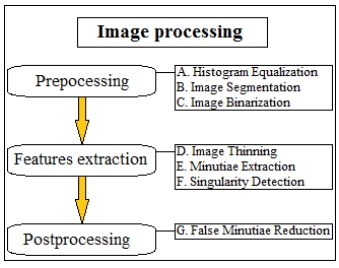


Fig. 3. Fingerprint's image processing steps.

3.1 Histogram equalization

Histogram equalization was chosen as the first step for the image enhancement process and in order to adjust the image's contrast. This method allows lower contrast regions of the image to gain a higher contrast.

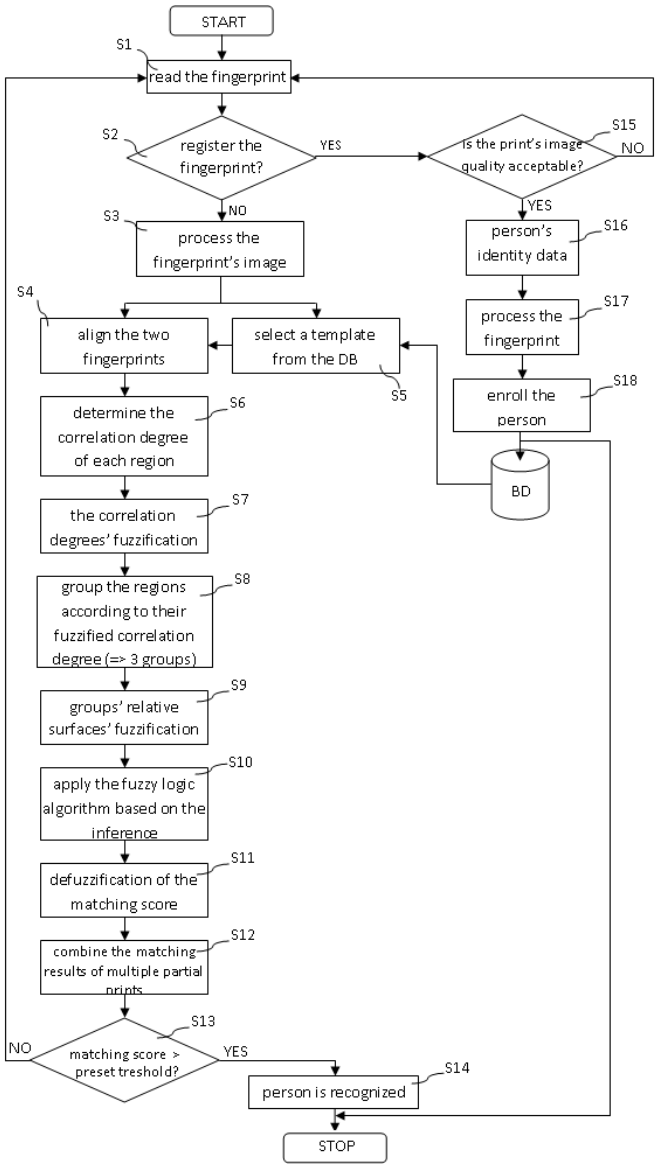


Fig. 2. The software architecture of the proposed system

The images' size captured with the optic sensor used to implement the recognition system presented in this paper is 256×320. The fingerprint's image is in the grayscale domain, so the intensity values range from 0 (black) to 255(white).

The image is divided into 32×32 pixels blocks. A simple local histogram equalization algorithm is performed for each block, in three steps:

1) Count the pixels of each intensity level (histogram formation).
2) Calculate the new intensity level for each of the 256 initial levels as in:

$$nil_i = \left[\sum_{j=0}^{i} N_j\right] * \frac{max.intensity\ level}{pixels\ no.}, \quad i=0,..,255, \quad (1)$$

where nil_i is the new intensity level, *max.intensity level* is 255, *pixels no.* for one block is 1024 and the expression in brackets represents the number of pixels from the initial image having a lower than *i* intensity level or equal to it.

3.2 Image segmentation

This step is performed in order to separate the fingerprint from the background, thus avoiding feature extraction from noisy areas. A common method is to use the local orientation and the local ridge frequency of the fingerprint.

Local orientation

In order to determine the local orientation the image is divided into 32×32 pixels blocks. According to Wang et al. (2007) the local orientation of a block (denoted by θ_B) can be calculated as in:

$$\theta_B = \frac{1}{2}\operatorname{atan}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} 2\,\partial_x(i,j)\,\partial_y(i,j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \partial_x^2(i,j) - \partial_y^2(i,j)} + \frac{\pi}{2}, \quad (2)$$

where N is the block size and $\partial_x(i,j)$ and $\partial_y(i,j)$ are the gradient magnitudes of the pixel located at (i,j) coordinates on the x and y directions respectively, and can be calculated by using the Sobel operators.

Local ridge frequency

The frequency of a block is determined as shown in Fig. 4.

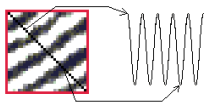


Fig. 4. Local ridge frequency

The intensity levels of the pixels found on the orthogonal direction through the block's center to its orientation are represented in a two dimensional planes. Ideally, the result has a sinusoidal shape. The local frequency can be determined by counting the sinusoid's negative peaks (Ratha et al. (1995)). The blocks that don't have a striped oriented pattern are removed, according to Maltoni (2005).

3.3 Image binarization

Image binarization transforms a greyscale 8-bit image into a 1-bit image (black ('0') and white ('1') only). For better results, the binarization process can be performed locally, meaning that a local threshold will be determined for each block.

3.4 Image thinning

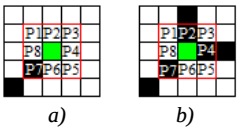
Thinning is a morphological operation that is used to reduce the ridges' thickness to one pixel. The algorithm that was used for this stage of the fingerprint image processing is proposed by MathWorks (2009). Fig. 5 c) shows the resulting thinned image.

a) b) c)

Fig. 5. a) Initial image; b) Segmented and binarized image; c) Thinned image

3.5 Minutiae extraction

After the thinning process, determining the minutiae is done by examining the 8-*neighberhood* of each ridge skeleton pixel, as shown in Fig. 6.



a) *b)*

Fig. 6. Minutiae extraction: a) termination, b) bifurcation

3.6 Singularity detection

Most singularity detection methods are based on the local ridge orientation image. The most well known method based on the Poincare index was proposed by Kawagoe (1984). This method does not work on poor quality images and fails to detect most arch-type fingerprints. Park et al. (2003) proposes to shift a rectangular window across the fingerprint image until the upper region of the window contains the most ridges having an orientation of 0^o (refer to Fig. 7). If a singularity exists, then the ridges' orientation from the lower region changes abruptly. The singular point is considered to be on the lowest ridge from the upper region.

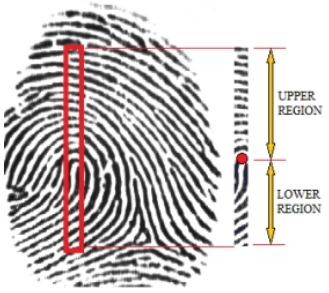


Fig. 7. Ridge distribution in the neighbourhood of a singularity

3.7 False minutiae reduction

There are a number of false minutiae that have to be removed from the fingerprint image. This step is necessary in order to eliminate noisy formations like those presented in Fig. 8.

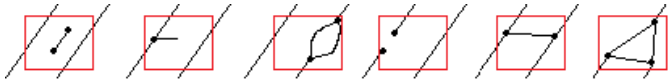


Fig. 8. False minutiae resulted after image thinning

4. PARTIAL FINGERPRINT MATCHING ALGORITHM

Direct application of correlation-based algorithms is computationally very expensive, due to the large number of rotations and translations needed. In order to reduce the computational time, the proposed algorithm tries to match two minutiae from the input with two minutiae from the template. After aligning the two minutiae sets the two images can be correlated.

Let I and T be the minutiae sets of the input and of the template fingerprints, respectively. Each minutia m is denoted by a triplet {x, y, θ}, where x and y are the location's coordinates and θ is the minutia's angle.

$I= \{m_1, m_2, ..., m_k\}$, $m_i=\{x_i, y_i, \theta_i\}$, i=1..k,

$T= \{m'_1, m'_2, ..., m'_l\}$, $m'_j=\{x_j, y_j, \theta_j\}$, j=1..l, (4)

where k and l are the numbers of minutiae in I and T, respectively.

Both I and T sets are divided into two subsets, each subset containing only the termination or bifurcation minutiae:

$I= I_T \cup I_B$,

$T= T_T \cup T_B$, (5)

where I_T and I_B denote the input's termination and bifurcation subsets, and T_T and T_B denote the template's termination and bifurcation subsets, respectively.

The proposed algorithm tries to find two bifurcation minutiae in the partial input fingerprint at a maximum distance from each other and matches them to two minutiae from the template. Bifurcation minutiae are considered to be more reliable because false terminations can be introduced by the image capturing or image processing stages. If the input presents no bifurcations ($I_B=\emptyset$) then two terminations are chosen. When there is only one bifurcation then that one and another termination are considered.

The distance between two minutiae is calculated with the formula:

$$dist(m_a, m_b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \quad (6)$$

The following pseudocode presents the minutiae correlation algorithm:

Algorithm 1: Minutiae correlation algorithm

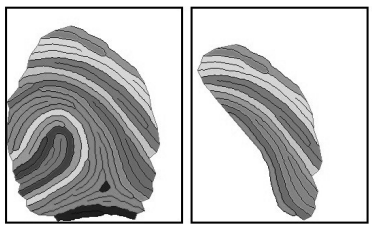
input: I_T, I_B;
input: T_T, T_B;
output: (m_a, m_b), (m'_c, m'_d);//minutiae pairs from the input and the template, respectively.

//determine the two input minutiae
case of cardinal(I_B)
  0: choose $m_a \in I_T$ **and** $m_b \in I_T$ | *dist*(m_a, m_b)= max;
    $M_c=T_T$; $M_d=T_T$;
  1: choose $m_a \in I_B$ **and** $m_b \in I_T$ | *dist*(m_a, m_b)= max;
    $M_c=T_B$; $M_d=T_T$;
  >2: choose $m_a \in I_B$ **and** $m_b \in I_B$ | *dist*(m_a, m_b)= max;
    $M_c=T_B$; $M_d=T_B$;
end case;

//determine the corresponding template minutiae
distance=*dist*(m_a, m_b); //
m'_c:=0; m'_d:=0;
for each $m'_i \in M_c$ **do**
  for each $m'_j \in M_d$ **do**
    if (distance ≈ *dist*(m'_i, m'_j)) **then**
      θ_{diff} :=θ_a-θ'_i; //θ_a, θ'_i angles of m_a, m'_i
      if ($\theta'_j \approx (\theta_b+\theta_{diff})$) **then**
        m'_c:= m'_i; m'_d:= m'_j;
      end if;
    end if;
  end for;
end for;
return (m_a, m_b), (m'_c, m'_d);

Another important aspect is that the proposed algorithm correlates the regions between the thinned ridges, not the ridges themselves. Choosing to correlate the regions between the ridges is a good idea, since due to non-linear distortions and image processing stage; the thinned ridges from the input and the template can be slightly shifted and/or rotated, even if the images were taken from the same fingerprint.

In order to compare the input to the template a region coloring step is required for the both image. First, the fingerprint regions have to be enclosed from the background. This is done by uniting the neighbor ridge endings with straight lines. Second, all the regions are colored. This process is similarly to the use of Microsoft Paint's Fill with color Tool and it is performed by a labelling algorithm. Both the input and the template are colored as presented in Fig. 9.



a) b)

Fig. 9. a) Template region colouring; b) Input region colouring

Let R_I and R_T be the input region sets and the template region set, respectively. These distinct regions are the result of the colouring process.

$R_I = \{r_i \mid i=1..m\}$,

$R_T = \{r'_j \mid j=1..n\}$, (7)

where m and n represent the numbers of different regions from the input and from the template, respectively.

The input image is shifted along Ox and Oy axes until m_a overlaps m_c ($x_a=x_c$ and $y_a=y_c$). The input image is rotated around the minutia location (x_a,y_a) until m_b overlaps m_d ($x_b=x_d$ and $y_b=y_d$). The two images are now considered to be aligned. The correlation degree of each region is determined (refer to Algorithm 2).

Algorithm 2: Correlation degree algorithm

input: imI, imT; //the two images: input, template
input: m, n; //the numbers of regions in I and T
output: $op_i \mid i=1..m$; //maximum number of overlapping pixels
for each $r_i \in R_I$ **do**
  //reset the auxiliary vector *aux_op*
  for i:=1 **to** n **do**
    aux_op_i :=0;
  end for;
  //increment the auxiliary vector's element corresponding to i^{th} region
  for each imI(j) pixel from r_i **do**
    h:=*getRegionIndexForPixel*(imI(j));
    aux_op_h ++;
  end for;
  op_i :=max(aux_op);
end for;
return op;

Algorithm 2 determines the maximum number of overlapping pixels of each region from R_I onto a single region from R_T, after the two images were aligned (denoted by the vector op_i, i=1..m). The correlation degree of each region denoted by cd_i is calculated with the formula:

$$cd_i = \frac{cp_i}{tp_i}\ [\%], i = 1..m, \quad (8)$$

where tp_i denotes the total number of pixels of each region r_i from R_I.

4.1 Fuzzification

In order to determine the correlation degree between the partial input fingerprint and the template, fuzzy logic rules are used.

The correlation degrees of each input region are fuzzified by using the well known membership function presented in Fig. 10. The fuzzified variables denoted by cdf_i (i=1..m) take values in the domain {L, M, H} (i.e. low, medium and high).

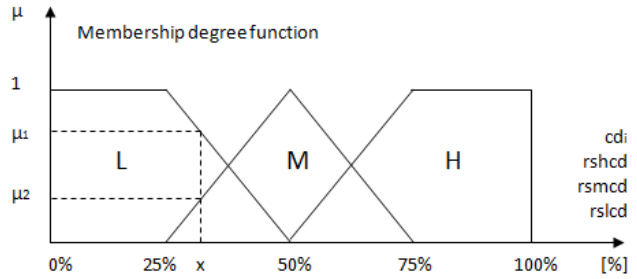


Fig. 10. Membership function

Let hcd_j, mcd_k and lcd_l be the sets of high correlation degree, medium correlation degree and low correlation degree regions, respectively.

$$hcd_j = H, \forall j = 1..\alpha,$$
$$mcd_k = M, \forall k = 1..\beta,$$
$$lcd_l = L, \forall l = 1..\gamma,$$
$$\alpha + \beta + \gamma = m. \quad (9)$$

The relative surfaces of each of the three set described above are calculated:

$$rshcd = \sum_{j=1}^{\alpha} rs_j,$$
$$rsmcd = \sum_{k=1}^{\beta} rs_k,$$
$$rslcd = \sum_{l=1}^{\gamma} rs_l. \quad (10)$$

where *rshcd*, *rsmcd* and *rslcd* are the relative surfaces of: all the high correlation degree regions, all the medium correlation degree regions and all the low correlation degree regions, respectively, and rs_i is the relative surface of the i^{th} region.

These three parameters are also fuzzified in the domain {L, M, H} by using the same membership function presented in Fig. 10, resulting the fuzzy variables: *rshcdf*, *rsmcdf* and *rslcdf*.

The proposed fuzzy matching algorithm is based on the inference and uses rules with the following form:

IF (*rshcdf* is X) AND (*rsmcdf* is Y) AND (*rslcdf* is Z) THEN (*matchf* is W)

where X, Y, Z and W take values in the set {L, M, H}.

The fuzzy logic rules described above are implemented as shown in Table 1, Table 2 and Table 3. These tables provide the fuzzy logic values for the output variable *matchf*.

Table 1. Fuzzy logic rules (rshcdf is H)

rshcdf is H		*rsmcdf*		
		H	**M**	**L**
rslcdf	**H**	-	-	H
	M	-	H	H
	L	H	H	H

Table 2. Fuzzy logic rules (rshcdf is M)

rshcdf is M		*rsmcdf*		
		H	**M**	**L**
rslcdf	**H**	-	M	M
	M	M	M	M
	L	M	M	M

Table 3. Fuzzy logic rules (rshcdf is L)

rshcdf is L		*rsmcdf*		
		H	**M**	**L**
rslcdf	**H**	M	L	L
	M	M	L	L
	L	M	L	L

The table cells that contain no fuzzy values represent the rules which cannot be activated (e.g. *rshcd*+*rsmcd*+*rslcd*>100%).

4.2 Defuzzification

Due to the fact that more than one rule can be activated for the same set of input crisp values (because each fuzzy variable can have two membership grades as shown in Fig. 10), the defuzzification process involves the use of the rule *strengths*. Only rules that have medium and high outputs influence the final matching score. The strength of each rule is calculated as in (11). The weight of the *rshcdf* membership grade is twice more important than the weight of *rsmcdf* membership grade Cherrak et al. (1998).

$$Strength(rule_i) = \frac{2}{3}\mu_k(rshcdf) + \frac{1}{3}\mu_k(rsmcdf). \quad (11)$$

where $\mu_k(x)$ is the membership grade function and i is the number of the activated rule.

Exception for (11): If *rshcd=100%* then s_i=1.

The logical products for each rule must be combined (or inferred) before performing the defuzzification process that results in crisp output values.

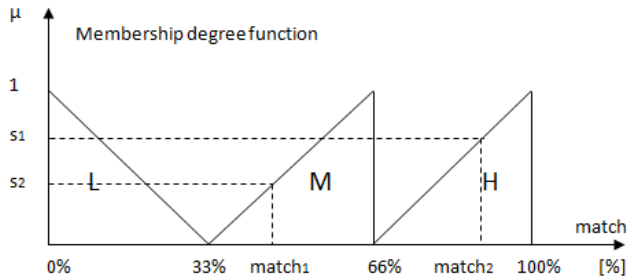


Fig. 11. Output membership function

Fig. 11 depicts the defuzzification function of the fuzzy output *matchf*. The L, M, H intervals' bounds were

experimentally set. Raising the lower bounds of M and H intervals means a higher security level (or higher rejection rate). Employing genetic algorithms to determine these bounds was considered as future work.

The crisp values of the output variable *matchf* are calculated by the centre of gravity for singleton method as in (12):

$$match\,[\%] = \frac{\sum_{i=1}^{t} match_i * s_i}{\sum_{i=1}^{t} s_i}, \quad (12)$$

where s_i is the strength of the i[th] rule and t is the number of the activated rules for the same input set.

4.3 Combining the matching results of multiple partial prints

Provided that there are n partial prints of the same fingerprint, the proposed algorithm can calculate a combined matching score based on the fuzzy logic algorithm presented above.

The crisp values of each partial fingerprint matching score are denoted by $match_i$ (i=1..n). The ratio between the partial input fingerprint surface and the total surface of the template (in pixels) is denoted by $relativeS_i$ (i=1..n).

Before the total matching score is calculated, the overlapping areas of the partial fingerprints must be taken into account.

If two partial fingerprints overlap each other the print that has a lower matching score yields the overlapping area.

The total matching score (denoted by *matchT*) is calculated with the next formula:

$$matchT = \sum_{i=1}^{n} match_i * relativeS_i, \quad (13)$$

6. CONCLUSIONS

A fuzzy logic based algorithm that involves the correlation of the regions enclosed by the thinned ridges of a partial fingerprint input with those from the template is proposed for matching partial fingerprints. This algorithm combines the temporal performances of the minutiae based algorithms with the reliability of the correlation based ones.

The algorithm proposed for matching partial fingerprints can be used to restrict the access to low or medium security resources. The access to such resources might be granted based on the combined results of successive authentication attempts during which only partial fingerprints are captured, not necessarily requiring a good quality input image.

Further improvements of the proposed system involve the use of genetic algorithms for tuning the defuzzification parameters and the establishment of a partial fingerprint database for performance comparisons.

The system's modules were implemented in Java 2 SE, but at the moment they are not fully integrated. The system will be tested on a large fingerprint database as soon as the modules' integration is completed. The image processing modules were also implemented in Matlab 7.8 as a prototype. Table 4 presents a temporal performance comparison between the execution times of the same modules under the two environments.

Table 4. Temporal performances

	Segmentation [ms]	Histogram Equalization [ms]	Binarization [ms]	Thinning [ms]
Matlab 7.8	166	7	10	23
Java SE 1.6	1120	20	21	520

REFERENCES

Cherrak, I., Jaulent, M.C., Degoulet, P. (1998). *A Fuzzy Classification System to Predict Renal Artery Restenosis after Angioplasty. Procedings AMIA Symp.*, p. 582–586.

Henry, E.R. (1900). *Classification and Uses of Fingerprints.* Routledge, London.

International Biometric Group (2009), *Biometrics Market and Industry Report 2009-2014*, p. 18-37.

Kawagoe, M., and Tojo, A. (1984). *Fingerprint Pattern Classification. Pattern Recognition,* volume 17, p. 295−303.

Le, H., and Bui, D. (2009). *Online fingerprint identification with a fast and distortion tolerant hashing.* Journal of Information Assurance and Security 4, p. 117-123.

Leung, W.F., Leung, S.H., Lau, W.H., Luk, A. (1991). *Fingerprint Recognition Using Neural Network,* Neural Networks For Signal Processing – Proceedings Of The 1991 IEEE Workshop.

Maltoni, D. (2005). *A Tutorial on Fingerprint Recognition.* Biometric Systems Laboratory - DEIS - University of Bologna.

Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S. (2003). *Handbook of Fingerprint Recognition.* p 3-7. Springer, New York.

Marana, A.N., Jain, A.K. (2005). *Ridge-Based Fingerprint Matching Using Hough Transform.* Computer Graphics and Image Processing, SIBGRAPI, p. 112-119.

Park, C.H., Oh, S.K., Kwak, D.M., Kim, B.S. (2003). *A New Reference Point Detection Algorithm Based on Orientation Pattern Labelling in Fingerprint Images.* Pattern Recognition and Image Analysis, IbPRIA, 2003.

Ratha, N.K., Chen, S.Y., Jain, A.K. (1995). Adaptive Flow Orientation-Based Feature Extraction in Fingerprint Images. *Pattern Recognition,* volume 28, no. 11, p. 1657−1672.

The MathWorks, Inc., (2009). *Image Processing Toolbox User's Guide.*

Wang, Y., Hu, J., Han, F. (2007). *Enhanced gradient-based algorithm for the estimation of fingerprint orientation fields.* Applied Mathematics and Computation 185, p. 823–833.

Wegstein, J.H., (1982). *An Automated Fingerprint Identification System*. U.S. Government Publication, Washington, DC: U.S. Dept. of Commerce, National Bureau of Standards.

Zhang, W., and Wang, Y. (2002). *Core-Based Structure Matching Algorithm of Fingerprint Verification*. Pattern Recognition. Proceedings of the 16th International Conference on, IEEE, p. 70-74.