A Fast Fingerprint Identification Approach via Circular String Matching Filters

Mai Alzamel
Department of Informatics
King's Collage London
Email:mai.alzamel@kcl.ac.uk

Moudhi Aljamea
Department of Informatics
King's Collage London
Email: mudhi.aljamea@kcl.ac.uk

Costas S. Iliopoulos
Department of Informatics
King's Collage London
Email:c.iliopoulos@kcl.ac.uk

M . Samiruzzaman
Department of Informatics
King's Collage London
Email:mohammad.samiruzzaman@kcl.ac.uk

***Abstract*—Fingerprints are used in various fields for different purposes. A typical fingerprint systems database may consist of millions fingerprint templates. The performance of matching with a large database has been the focus of in the recent research. This paper proposes an efficient database indexing technique using four fast and lightweight effective filters in a circular pattern matching technique [1]. This approach reduces the number of nominated fingerprints before the matching stage by applying the filters on the database gradually. The filtering algorithms used here are robust and fast and are performed in linear time regarding the fingerprints' rotation. The entire approach has been explained in detail in section V. This paper is an extension of the recent paper [2]. The main motivation of the extension is to deal with fingerprint authentication for one to many fingerprint authentication approaches efficiently.**

***Keywords*—Fingerprints; database; indexing; filters; identification**

I. Introduction

Automated Fingerprint Recognition Systems (AFRS) are usually deployed in two modes: verification and identification. Furthermore, these systems can be found in many application, such as managing employees' attendance, border control, access control to sensitive data and so on. The verification recognition mode is a one-to-one matching procedure that matches the scanned fingerprint to the same registered fingerprint in the database [3]. The identification recognition mode, meanwhile, is a one-to-many matching procedure that identifies only a number of fingerprints from which are saved in the database and matches the scanned fingerprint [3]. Identification recognition mode databases may contain a large volume of data due to the system requirements, for instance, immigration systems deal with millions of saved fingerprints daily.

There are many techniques to process these large volumes of data, but the most popular is indexing the fingerprint database to adopt a subset of nominated fingerprints. There are two main classes in fingerprint indexing: the first uses features called ridge orientation maps and ridge frequency maps [4], [5] whereas, the second uses features called minutiae. Generally, the first class is faster than the second one since it shows the feature in compact vectors. On the other hand, the second class is more accurate since the minutiae contain more distinguishable information about the fingerprints than ridge orientation and frequency maps.

Moreover, combining both indexing approaches has been attempted to improve the performance regarding speed and accuracy [6], [7], but since, the matching process speed is affected by the accuracy level, achieving higher accuracy will result in a slower the matching process [3], [8].

II. Background

Human fingerprints can be represented as a composite of ridges and grooves on the fingertip skin. The human fingertip skin consists of minute ridges, and furrows between each ridge, these two combine to form a sophisticated pattern by running the ridges and furrows in parallel lines and curves. [9] mentioned three main fingerprint patterns, which are whorl, loop and arch. In addition, Henry's fingerprint classification system classified fingerprints into eight classes: Plain Arch, Tented Arch, Left Slant Loop, Right Slant Loop, Plain Whorl, Double-Loop, Whorl Central-Pocket Whorl and Accidental Whorl [10], [11].

Every fingerprint is highly immutable and unique, this uniqueness is defined by minutiae which can be explained as the combinations of local features such as (ridge endings and ridge bifurcations), global features such as (ridges and valleys) and by local features [12].

In fact, fingerprints are an old tool for identifying people, adopting the so-called ink-technique [3]. This technique involves dabbing the fingers with ink to print the fingerprints on paper which can later be scanned to create a digital record of the fingerprints. This so-called off-line fingerprint approach matches fingerprints by utilising the achieved digital images. Despite the importance of the above technique in the field of forensics it is not applicable for biometrics applications [13]. The most recent techniques aim to achieve to get the matched fingerprints in real time.

III. Our Contribution

In this paper, we illustrate the fingerprint recognition issue that rely on any fingerprint verification and identification system. Although, the fingerprint identification problem in the

string matching field is not a new topic, there is still room for improvment [14].

Interestingly enough, our proposed algorithm in [15] was the first attempt to utilise circular string matching techniques to solve fingerprints recognitions problems accurately and efficiently. The proposed algorithm was to convert the fingerprints image into circular strings mainly to solve the rotation problem. The matching stage was done in linear time by comparing the string extracted from the input fingerprint image with the stored image which is saved in a string format in the database.

As mentioned previously, the algorithm in [15] focused on solving the fingerprint problem efficintly regardless of the scanned fingerprint poistion and degree of rotation. Later, in [2] we proved the high accuracy and efficiency of the algorithm by implementing it and analysing the experiment results.

In this paper we extend our method so it can be used in an identification mode with a large database. Accordingly, our focus is to reduce the database space complexity before the matching stage.

A. Road Map

The organization of the rest of this paper is as follows. In Section II, we presented some background related to fingerprints, and then our contribution in III. In section IV we present a brief literature review of the related work. We describe our approach in Section V and discuss some preliminaries in Section VI. The filtering algorithms are set out in section VII and we briefly conclude in section VIII.

IV. Related works

When it comes to solving fingerprint identification problems with large databases, fingerprint indexing and fingerprint classification have been the most reliable solutions until now [16]. Henry's classification system [17] classified fingerprints into five categories: left loop, right loop, whorl, arch and tented arch. This number of categories, however, is relatively small compared to the number of fingerprints in modern databases. To explain, fingerprint indexing represents the fingerprints with feature vectors. This approach is deployed at the matching stage where a query fingerprint will be matched with those fingerprints in the database which have vectors similar to the query vector [16]. Various algorithms have been proposed to improve fingerprint indexing, involving either global features or minutiae features.

1) Minutia Feature: The main concept is to struct a robust and discriminant minutia with rotation and translation problems. In [18] and [19] the proposed algorithms based on features of triangle or quadrangle geometric and selected simple hashing methods. However, these geometric characteristic has more sensitivity to distortion and noise. whereas, [20] designed a descriptor used a minutiae triplet, their features of triplet consists the angles, the length of each length and the ridge to be calculated between each minutiae pair. However, a false rate might be happening due to that the minutia is superior in discrimination. A pairwise enrolment is used to reduce the false rate between each input fingerprint and the stored fingerprints in the database using clustering transformation parameter, however, this costs a lot of time.

Later, [18] developed the algorithm in [20] to increase the matching accuracy and the speed. They used a new minutiae triplets feature and effective ad-hoc geometric rules on the theory of triplets matching instead of pairwise enrollment. Therefore, an logarithm has been proposed, in [21] to use a binary Minutia Cylinder Code (MCC) to be a minutia descriptor. This approach is proposed to encode the direction and location of neighbour minutiae per minutia into a bit vector with is fixed length. Although MCC has higher accuracy and speed it requires a higher dimension. Alternatively, Locality Sensitive Hashing (LSH) is utilised to search for MCC hypothesis correspondences rather than the original quantisation strategy.This is not efficient, however for applications that need high accuracy.

2) Global Feature: The main idea of global feature is that ridges information are represented by specific points or orientation domains. In [22] and [23] select particular points for use as features in fingerprints indexing. The drawbacks of this method are that it needs fingerprint pre-alignment and that it is challenging to extract certain points from specified locations within from distorted images. Moreover, [24] and [4] illustrated fingerprint indexing derived from orientation domain to represent the fingerprint feature vector. The main problem of these proposed features is that they are global and thus are not readable to distinguish the minutiae.

V. Our Approach

In this paper we present a new technique to reduce the number of fingerprints that will be matched in the database by applying a linear time filter on the saved fingerprints regardless of the different rotations of these fingerprints in the database, that will be done as a pre-processing step to the fingerprint matching stage.

The fingerprints in the database will be saved in a circular string format that will be extracted using our extraction algorithm in [15] as shown in algorithm 1. Afterwords, the four filters that were proposed in [1] will be applied to the database and the result will be saved for each fingerprint. Then, during the identification process, the filters will be applied on the scanned fingerprint as explained in algorithm 2. Just by filtering, therefore, a large number of circular string patterns will be reduced to a handful of rows in the database. The reduction in search space has been shown experimentally in [25] of circular patterns. Algorithm 2 considers that the pattern consists four letters A,C,G and T, and then converts them to a specific digit to deploy the four observations. In our paper,since the patten consists only 0s or 1s we will skip this step.I illustrates more how we apply the 4 filters on the pattern.

A. Problem Definition

Here we consider the problem of finding occurrences of a pattern string $\mathcal{P}$ of length m with filters circular structure in a text string $\mathcal{T}$ of length n with linear circular structure.

For instance, in the binary sequence of a circular structure, if we want to find occurrences of a particular string in a text sequence, which may not be circular, it must locate all positions in $\mathcal{T}$ where at least one rotation of $\mathcal{P}$ occurs. This is the problem of *circular pattern matching* (CPM).

Algorithm 1 Novel Minutiae Extraction [2]

procedure $novel\_minutiae\_extraction(img[][], r, cx, cy)$
  "img", a 2d char array representing a fingerprint scan
  "r", the radius of the current circular scan
  "(cx, cy)", the coordinates of the center of the current circular scan
  "$pattern$", a circular binary string
  for $i \leftarrow cx - r$ to $cx + r$ **do**
    for $j \leftarrow cy - r$ to $cy + r$ **do**
      if $r^2 = (i - x)^2$ to $(i - y)^2$ **then**
        if $x < cx$ **then**
          if $img[i][j] < 125$ **then**
            append 0 to the left of pattern
          else
            append 1 to the left of pattern
          end if
        else
          if $img[i][j] < 125$ **then**
            append 0 to the right of pattern
          else
            append 1 to the right of pattern
          end if
        end if
      end if
    end for
  end for
  return pattern
end procedure

Algorithm 2 Approximate Circular Pattern Signature using Filters $1:4$ [1]

procedure $ACPS\_FT(\mathcal{P}[1:m])$
  define three variables for observations 1, 2, 3
  define an array of size 4 for observation 4
  define an array of size 4 to keep a fixed value of A, C, G, T
  $s \leftarrow \mathcal{P}[1:m]\mathcal{P}[1]$
  initialize all defined variables to zero
  initialize the fixed array to $\{1, 2, 3, 4\}$
  for $i \leftarrow 1$ to $|s|$ **do**
    if $i \neq |s|$ **then**
      calculate different filtering values via observations 1 & 4 and make a running sum
    end if
    calculate different filtering values via observations 2, 3 and make a running sum
  end for
  return all observation values
end procedure

VI. Preliminaries

Let Σ be a finite *alphabet*. An element of Σ^* is called a *string*. The length of a string w is denoted by $|w|$. The empty string ϵ is a string of length 0, that is, $|\epsilon| = 0$. Let $\Sigma^+ = \Sigma^* - \{\epsilon\}$. For a string $w = xyz$, x, y and z are called a *prefix*, *factor* (or equivalently, *substring*), and *suffix* of w, respectively. The i-th character of a string w is denoted by $w[i]$ for $1 \leq i \leq |w|$, and the factor of a string w that begins at position i and ends at position j is denoted by $w[i:j]$ for $1 \leq i \leq j \leq |w|$. For convenience, we assume $w[i:j] = \epsilon$ if $j < i$. A k-factor is a factor of length k.

A circular string of length m can be viewed as a traditional linear string which has the left-most and right-most symbols wrapped around and stuck together in some way. Under this notion, the same circular string can be seen as m different linear strings, which would all be considered equivalent. Given a string $\mathcal{P}$ of length m, which 33we denote by $\mathcal{P}^i = \mathcal{P}[i:m]\mathcal{P}[1:i-1]$, $0 < i < m$, and the i-th *rotation* of $\mathcal{P}$ and $\mathcal{P}^0 = \mathcal{P}$.

1) Example: Suppose we have a pattern $\mathcal{P} = atcgatg$. The pattern $\mathcal{P}$ has the following rotations (i.e., conjugates): $\mathcal{P}^1 = tcgatga, \mathcal{P}^2 = cgatgat, \mathcal{P}^3 = gatgatc, \mathcal{P}^4 = atgatcg, \mathcal{P}^5 = tgatcga, \mathcal{P}^6 = gatcgat$.

The *Hamming distance* between strings $\mathcal{P}$ and $\mathcal{T}$, both of length n, is the number of positions i, $0 \leq i < n$, such that $\mathcal{P}[i] \neq \mathcal{T}[i]$. Given a non-negative integer k, we write $\mathcal{P} \equiv_k \mathcal{T}$ or equivalently say that $\mathcal{P}$ k-matches $\mathcal{T}$, provided that the Hamming distance between $\mathcal{P}$ and $\mathcal{T}$ is at most k.

We consider the binary alphabet, i.e., $\Sigma = \{0, 1\}$. In our approach, each character of the alphabet is presented by its numeric value. We use $num(x), x \in \Sigma$ to denote the numeric value of the character x. So, we have $num(0) = 0, num(1) = 1$. For a string S, we use the notation S_N to denote the numeric representation of the string S; and $S_N[i]$ denotes the numeric value of the character $S[i]$. So, if $S[i] = 1$ then $S_N[i] = num(1) = 1$. The concept of the circular string and their rotations also applies naturally to their numeric representations as is illustrated in the example below.

2) Example: Suppose we have a pattern $\mathcal{P} = 0110101$. The numeric representation of $\mathcal{P}$ is $\mathcal{P}_N = 0110101$. And this numeric representation has the following rotations: $\mathcal{P}_N^1 = 1101010$, $\mathcal{P}_N^2 = 1010101$, $\mathcal{P}_N^3 = 0101011$, $\mathcal{P}_N^4 = 1010110$, $\mathcal{P}_N^5 = 0101101$, $\mathcal{P}_N^6 = 1011010$, $\mathcal{P}_N^7 = 0110101$.

The problem we handle in this article can be formally defined as follows.

3) problem: (Approximate Circular Pattern Matching with k-mismatches (ACPM)). Given a pattern $\mathcal{P}$ of length m, a text $\mathcal{T}$ of length $n > m$, and an integer threshold $k < m$, find all factors $\mathcal{F}$ of $\mathcal{T}$ such that $\mathcal{F} \equiv_k \mathcal{P}^i$ for some $0 \leq i < m$. And when we have a factor $\mathcal{F} = \mathcal{T}[j : j + |\mathcal{F}| - 1]$ such that $\mathcal{F} \equiv_k \mathcal{P}^i$ we say that the circular pattern $\mathcal{C}(\mathcal{P})$ k-matches $\mathcal{T}$ at position j. We also say that this k-match is due to $\mathcal{P}^i$, i.e., the ith rotation of $\mathcal{P}$.

VII. Filtering Algorithm

As has been mentioned above, the algorithm is based on some filtering techniques. Suppose we are given a pattern $\mathcal{P}$

and a text $\mathcal{T}$. We will frequently and conveniently use the expression "$\mathcal{C}(\mathcal{P})$ k-matches $\mathcal{T}$ at position i" (or equivalently, "$\mathcal{P}$ circularly k-matches $\mathcal{T}$ at position i") to indicate that one of the conjugates of $\mathcal{P}$ k-matches $\mathcal{T}$ at position i (or equivalently, $\mathcal{C}(\mathcal{P}) \equiv_k \mathcal{T}$). We start with an brief overview of the filters below.

A. Database Filters

We employ a total of four filters from [1] to solve the one-to-many fingerprint problem. We are giving an overview of the filters here. For the detailed explanation and results to [1]. The key to the observations and the resulting filters is the fact that each function has devised results in a unique output when applied to the rotations of a circular string.

1) Filter 1: [1] defines the function sum on a string $\mathcal{P}$ of length m as follows: $sum(\mathcal{P}) = \sum_{i=1}^{m} P_N[i]$. Our first filter, Filter 1, is based on this sum function of [1]. We have the following observation.

Observation 1: Consider a circular string $\mathcal{P}$ and a linear string $\mathcal{T}$, both having length n. If $\mathcal{C}(\mathcal{P}) \equiv_k \mathcal{T}$, where $0 \leq k < n$, then we must have

$$sum(\mathcal{T}) - k \times 4 + k \times 1 \leq sum(\mathcal{C}(\mathcal{P})) \leq sum(\mathcal{T}) + k \times 4 - k \times 1.$$

2) Filters 2 and 3: [1] defines the second and third filters, i.e., Filters 2 and 3, depend on a notion of distance between consecutive characters of a string. The *distance* between two consecutive characters of a string $\mathcal{P}$ of length m is defined by $distance(\mathcal{P}[i], \mathcal{P}[i+1]) = \mathcal{P}_{\mathcal{N}}[i] - \mathcal{P}_{\mathcal{N}}[i+1]$, where $1 \leq i \leq m-1$. We define $total\_distance(P) = \sum_{i=1}^{m-1} distance(\mathcal{P}[i], \mathcal{P}[i+1])$. We also define an absolute version of it: $abs\_total\_distance(P) = \sum_{i=1}^{m-1} abs(distance(\mathcal{P}[i], \mathcal{P}[i+1]))$, where $abs(x)$ returns the magnitude of x ignoring the sign. Before we apply these two functions on our strings to get our filters, we need to do a simple pre-processing on the respective string, i.e., $\mathcal{P}$ in this case, as follows. We extend the string $\mathcal{P}$ by concatenating the first character of $\mathcal{P}$ at its end. We use $ext(\mathcal{P})$ to denote the resultant string. So, we have $ext(\mathcal{P}) = \mathcal{P}\mathcal{P}[1]$. Since, $ext(\mathcal{P})$ can simply be treated as another string, we can easily extend the notation and concept of $\mathcal{C}(\mathcal{P})$ over $ext(\mathcal{P})$ and we continue to abuse the notation a bit for the sake of conciseness

Now we have the following observation which is the basis of our Filter 2.

Observation 2: Consider a circular string $\mathcal{P}$ and a linear string $\mathcal{T}$, both having length n, and assume that $\mathcal{A} = ext(\mathcal{P})$ and $\mathcal{B} = ext(\mathcal{T})$. If $\mathcal{C}(\mathcal{P}) \equiv_k \mathcal{T}$, where $0 \leq k < n$, then we must have

$$abs\_total\_distance(\mathcal{B}) - k \times 4 + k \times 1 \leq abs\_total\_distance(\mathcal{C}(\mathcal{A})) \leq abs\_total\_distance(\mathcal{B}) + k \times 4 - k \times 1.$$

Now [1] presents the following related observation which is the basis of our Filter 3. Note that Observation 2 differs from Observation 3 only through using the absolute version of the function used in the latter.

Observation 3: Consider a circular string $\mathcal{P}$ and a linear string $\mathcal{T}$, both having length n, and assume that $\mathcal{A} = ext(\mathcal{P})$ and $\mathcal{B} = ext(\mathcal{T})$. If $\mathcal{C}(\mathcal{P}) \equiv_k \mathcal{T}$, where $0 \leq k < n$, then we must have

$$total\_distance(\mathcal{B}) - k \times 4 + k \times 1 \leq total\_distance(\mathcal{C}(\mathcal{A})) \leq total\_distance(\mathcal{B}) + k \times 4 - k \times 1.$$

3) Filter 4: Filter 4 of [1] uses the $sum()$ function used by Filter 1, albeit, in a slightly different way. In particular, it applies the $sum()$ function on individual characters. So, for $x \in \Sigma$ we define $sum_x(\mathcal{P}) = \sum_{1 \leq i \leq |\mathcal{P}|, \mathcal{P}[i]=x} P_N[i]$. Now we have the following observation.

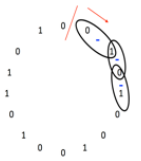
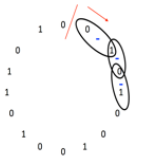
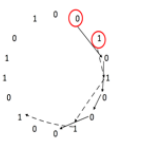
Observation 4: Consider a circular string $\mathcal{P}$ and a linear string $\mathcal{T}$ both having length n. If $\mathcal{C}(\mathcal{P}) \equiv_k \mathcal{T}$, where $0 \leq k < n$, then we must have

$$sum_x(\mathcal{T}) - k \times num(x) \leq sum_x(\mathcal{C}(\mathcal{P})) \leq sum_x(\mathcal{T}) + k \times num(x)$$

for all $x \in \Sigma$.

For clarity, the four filters are illustrated in the below table:

TABLE I. THE FOUR FINGERPRINT DATABASE FILTERS - IN RESPECT TO THE CIRCULAR STRING FORMAT

Circular Binary Image	Binary String p
	$Sum(p)$= 0+1+0+1+0+0+1+0+0+1+0+1+1+0+1+0=7
	$Sum(abs(distance(p)))$ = |(0-1)|+|(1-0)|+|(0-1)|+|(1-0)|+|(0-0)|+|(0-1)|+|(1-0)|+|(0-0)|+|(0-1)|+|(1-0)|+|(0-1)|+|(1-1)|+|(1-0)|+|(0-1)|+|(1-0)|+|0-0|=12
	$Sum((distance(p)))$ = (0-1)+(1-0)+(0-1)+(1-0)+(0-0)+(0-1)+(1-0)+(0-0)+(0-1)+(1-0)+(0-1)+(1-1)+(1-0)+(0-1)+(1-0)+(0-0)=0
	$Sum(idistance(0))$ = 2+2+1+2+1+2+3+2+1=16 $Sum(idistance(1))$ = 2+3+3+2+1+2+3=16

B. Time and Space Complexity

The main concept of this approach is to solve the fingerprint identification problem with high accuracy and speed using circular string matching, regardless of the scanned fingerprint position and rotation on the scanner. , given that this is considered as a problem in fingerprint matching systems [15].

To explain, whenever the rotation degree of the scanned fingerprint is increased the speed and accuracy of the matching process will decrease. This problem is solved in our approach, since it can match the fingerprint with high accuracy and speed without being affected by the degree of rotation of scanned fingerprint as shown in [2] . As an improvement, in this paper the proposed approach can deal with large volume databases. The proposed problem has been tackled by a combination of the work reported in [15], [2] and [1]. The time complexity of work done on circular pattern matching on [15], [2] are linear. The work done by filters in [1] are also linear in time and space.The time and space complexity for this approach is therefore linear.

VIII. Conclusion

This paper has proposed a new fingerprint identification technique for fast and accurate fingerprint recognition. The presented approach emphasised reducing the search space of the circular pattern matching problem by deploying four effective lightweight filtering techniques in a linear time. The entire approach has been done in three main stages: circular string extraction, applying the proposed four filters and finally, the matching stage. It has been proved that each phase can be performed in a linear time. Consequently, the entire approach will be achieved in a linear time with a large number of database entries. Future work will include an implementation of this paper as well as a comparison of the results with other one-to-many fingerprint identification algorithms in respect to their accuracy and speed.

References

[1] M. A. R. Azim, C. S. Iliopoulos, M. S. Rahman, and M. Samiruzzaman, "A filter-based approach for approximate circular pattern matching," in *Bioinformatics Research and Applications*. Springer, 2015, pp. 24–35.

[2] O. Ajala, M. Aljamea, M. Alzamel, C. Iliopoulos, and Y. Strigini, "Fast fingerprint recognition using circular string pattern matching techniques," pp. 47–52, 2016.

[3] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[4] M. Liu, X. Jiang, and A. C. Kot, "Efficient fingerprint search based on database clustering," *Pattern Recognition*, vol. 40, no. 6, pp. 1793–1803, 2007.

[5] M. Liu and P.-T. Yap, "Invariant representation of orientation fields for fingerprint indexing," *Pattern Recognition*, vol. 45, no. 7, pp. 2532–2542, 2012.

[6] R. Cappelli and M. Ferrara, "A fingerprint retrieval system based on level-1 and level-2 features," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 465–10 478, 2012.

[7] A. A. Paulino, E. Liu, K. Cao, and A. K. Jain, "Latent fingerprint indexing: Fusion of level 1 and level 2 features," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 1–8.

[8] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, and J. M. Benítez, "Fast fingerprint identification for large databases," *Pattern Recognition*, vol. 47, no. 2, pp. 588–602, 2014.

[9] Q. Zhang and H. Yan, "Fingerprint classification based on extraction and analysis of singularities and pseudo ridges," *Pattern Recognition*, vol. 37, no. 11, pp. 2233–2243, 2004.

[10] E. R. Henry, *Classification and Uses of Finger Prints*. Routledge, 1900.

[11] H. C. Lee, R. Ramotowski, and R. E. Gaensslen, Eds., *Advances in Fingerprint Technology, Second Edition*. CRC Press, 2002.

[12] S. Sebastian, "Literature survey on automated person identification techniques," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 5, pp. 232–237, 2013.

[13] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Springer-Verlag, 2009.

[14] P. D. Gutierrez, M. Lastra, F. Herrera, and J. M. Benítez, "A high performance fingerprint matching system for large databases based on gpu," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 1, pp. 62–71, 2014.

[15] A.-J. Moudhi, T. Athar, C. S. Iliopoulos, S. P. Pissis, and M. S. Rahman, "A novel pattern matching approach for fingerprint-based authentication."

[16] C. Bai, T. Zhao, W. Wang, and M. Wu, "An efficient indexing scheme based on k-plet representation for fingerprint database," in *Intelligent Computing Theories and Methodologies*. Springer, 2015, pp. 247–257.

[17] E. R. Henry, *Classification and uses of finger prints*. HM Stationery Office, 1905.

[18] B. Bhanu and X. Tan, "Fingerprint indexing based on novel features of minutiae triplets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 616–622, 2003.

[19] O. Iloanusi, A. Gyaourova, and A. Ross, "Indexing fingerprints using minutiae quadruplets," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 127–133.

[20] R. S. Germain, A. Califano, and S. Colville, "Fingerprint matching using transformation parameter clustering," *Computing in Science & Engineering*, no. 4, pp. 42–49, 1997.

[21] R. Cappelli, M. Ferrara, and D. Maltoni, "Fingerprint indexing based on minutia cylinder-code," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 5, pp. 1051–1057, 2011.

[22] T. Liu, G. Zhu, C. Zhang, and P. Hao, "Fingerprint indexing based on singular point correlation," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3. IEEE, 2005, pp. II–293.

[23] M. Liu, X. Jiang, and A. C. Kot, "Fingerprint retrieval by complex filter responses," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1. IEEE, 2006, pp. 1042–1042.

[24] Y. Wang, J. Hu, and D. Phillips, "A fingerprint orientation model based on 2d fourier expansion (fomfe) and its application to singular-point detection and fingerprint indexing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 4, pp. 573–585, 2007.

[25] M. Samiruzzaman, C. S. Iliopoulos, M. S. Rahman, and M. Azim, "A simple, fast, filter-based algorithm for approximate circular pattern matching," 2016.