# ANALYSIS OF COMPUTATIONAL COMPLEXITY FOR HT-BASED FINGERPRINT ALIGNMENT ALGORITHMS ON JAVA CARD ENVIRONMENT

Cynthia S. Mlambo[1], Meshack B. Shabalala[1],
Fulufhelo V. Nelwamondo[1,2]

[1] Council for Scientific and Industrial Research, Pretoria, South Africa,
smlambo@csir.co.za, mshabalala@csir.co.za
[2] Department of Engineering, University of Johannesburg
fnelwamondo@csir.co.za

***ABSTRACT***

*In this paper, implementations of three Hough Transform based fingerprint alignment algorithms are analyzed with respect to time complexity on Java Card environment. Three algorithms are: Local Match Based Approach (LMBA), Discretized Rotation Based Approach (DRBA), and All Possible to Match Based Approach (APMBA). The aim of this paper is to present the complexity and implementations of existing work of one of the mostly used method of fingerprint alignment, in order that the complexity can be simplified or find the best algorithm with efficient complexity and implementation that can be easily implemented on Java Card environment for match on card. Efficiency involves the accuracy of the implementation, time taken to perform fingerprint alignment, memory required by the implementation and instruction operations required and used.*

***KEYWORDS***

*Fingerprint Alignment, Java Card, Hough Transform, Smart Cards, Time Complexity.*

## 1. INTRODUCTION

The Java Card Environment have a limited instruction sets, unlike other languages. The challenge is that Smart Card applications are increasing in the market as one of the mostly used technologies. Currently what is happening in the industry is that most of the applications are shifting from computer based and large applications into small portable applications that can function on smart Cards. The basic recent application is the use of identification and verification of an individual using the Smart Card; this involves fingerprint based recognition systems.

Fingerprint alignment is a process of superimposing two different features of fingerprints that are captured at different instances [1]. This process is important in identifying or verifying if two fingerprint features captured at different instances are from the same finger. This is because there is always rotation and translation of a finger during the process of capturing fingerprint features [1]-[3]. One of the mostly used methods of performing fingerprint alignment is based on Hough Transform. The Hough Transform (HT) based methods accumulate votes for the most occurring rotation ($\Delta\theta$) and translation ($\Delta x$, $\Delta y$) between two sets of fingerprint features that are to be

Dhinaharan Nagamalai et al. (Eds) : CCSEA, DKMP, AIFU, SEA - 2015
pp. 33–41, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.50204

matched. Since Java Card applications are one of the intelligent technologies that are increasingly used these days [4]-[5]. However, The Java Card Environment has limited computing resources, such as; instruction sets, memory space and power sources [6]. There is a need of analysis existing applications on how is their performance on Java Card technologies. The challenge is that when implementing algorithms on Java Card, it needs to be modified so that it can meet the specifications of the Java Card environment [6]. Therefore, in this paper, four implementations of Hough Transform based Fingerprint Alignment algorithms are analysed on Java Card. These algorithms are: Local Match Based Approach (LMBA), Discretized Rotation Based Approach (DRBA), and All Possible to Match Based Approach (APMBA).

The aim of this paper is to present the complexity and implementations of existing work of one of the mostly used method of fingerprint alignment. In addition, two questions are answered in this research, stated as: How much changes or modifications are required? What are the effects of those modifications on time complexity, memory and performance of the algorithm? So that the complexity can be simplified or find the best algorithm with efficient complexity and implementation that can be easily implemented on smart cards. Efficiency involves the accuracy of the implementation, time taken to perform fingerprint alignment, memory required by the implementation and instruction operations required and or performed.

Fingerprint features used in this paper are minutiae points because they require less memory as they are represented as points [7]. Minutiae points are where ridges in minutiae points ends and split, alternatively called ridge ending and ridge bifurcation, respectively. Each minutia is presented in three coordinate, x-coordinate, y-coordinate and the orientation of the ridge.

This paper is organised as follows, firstly in Section II is a brief description of the HT based fingerprint alignment algorithms. Section III is the comparison of studied algorithms. Finally, section IV is the conclusion.

## 2. DEFINING PROBLEM STATEMENT

In this section the challenge on implementing HT-based fingerprint alignment algorithms is explained from the specifications of the Java Card environment [8]. In general definition, Java Card technology enables programs written in the Java programming language to run on smart cards.

The Java instruction sets are too large to fit on a resource constrained device such as a smart card [9]. That is why the Java Card environment consist limited instruction sets and card commands, which affect programming style and implementation of Java or low-level programming algorithms. Features that are common in Java language and Java Card are:

- Small primitive data types (Boolean, byte, short, int),
- One-dimensional arrays,
- Packages, classes, interfaces,
- Object-oriented features, (inheritance, virtual methods, overloading, dynamic object creation, access scope, binding rules),
- Exceptions [8].

However, the Java Card does not support some features [10]. As a result, the implementation of the algorithms that requires the following features needs to be changed. Unsupported features are:

- Large primitive data types (long, double, float),
- Characters and strings,
- Multidimensional arrays,
- Dynamic class loading,

- Security manager,
- Threads,
- Object cloning,
- Garbage collection,
- Object serialization [8].

Having discussed these Java Card features, it is common for most HT-based algorithms to require for example, multidimensional arrays [11] – [15]. Therefore, the research presented in analyses each algorithm based on how many does it require.

## 3. HOUGH TRANSFORM-BASED METHODS

In this section all three analysed algorithms are explained in details with their functionality on how they determine alignment parameters (*AP*) for translation and rotation. Two sets of fingerprints are taken as inputs to the algorithms, input (*I*) and template (*T*) set. Where *I* is a set of minutiae points from the input minutiae and *T* is a set of minutiae points from the database which was captured and stored.

### 3.1. All Possible Matching Based Approach

The first stage in this approach is performed by considering all points as possible matches [16], [17]. Therefore, for each minutiae point in input *I* and template *T,* all points are paired and alignment parameters are computed. For each set of computed alignment parameters, a vote is added to the accumulator array. In the last stage, after all minutiae points are paired, the most highly voted alignment parameters are determined and deemed as the best parameters for alignment of *I* and *T* [16]. This algorithm is explained in Algorithm 1.

The motive on this approach is that since the aim of alignment process is to estimate corresponding pairs. Minutiae points are not checked if are corresponding but all points are considered, with the idea of the HT, state: All corresponding points between set *I* and *T* will accumulate similar transformation into accumulator array *A* [17].

**Pros**:

- Simplicity and ease of implementation
- The use of one-dimensional array makes easy implementation
- General complexity of O(n*m)

**Cons**:

- Very inefficient for large number of minutiae points when performing voting process.

**Algorithm 1:** An Alignment Algorithm for Improved HT-Based Approach

**Input:** Sets of minutiae points from $I$ and $T$, and size of an image $[MaxSizeX, MaxSizeY]$ and maximum resolution $MaxSize\theta$.
**Output:** Set of $(\Delta_x, \Delta_y, \Delta_\theta)$.
Initialize Hough space into three 3D array
$ASize = sqrt(NumberofT * NumberofI)$
$BinSizeX = MaxSizeY/ASize$
$BinSizeY = MaxSizeX/ASize$
$BinSize\theta = MaxSize\theta/ASize$
Discretize values for $(\Delta_x, \Delta_y, \Delta_\theta)$ by initializing
$A_x[ASize]$,$A_y[ASize]$,$A_\theta[ASize]$
**for** *each minutiae* $m_t$ *in* $T$ **do**
  **for** *each minutiae* $m_i$ *in* $I$ **do**
    Compute direction difference $\theta^+$ between $(\theta_i)$ and $(\theta_t)$
    $\theta^+ = \min(|\theta_i - \theta_t|, 360 - |\theta_i - \theta_t|)$
    Compute translation parameters

$$\begin{bmatrix} \Delta_x^+ \\ \Delta_y^+ \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \cos\theta^+ & -\sin\theta^+ \\ \sin\theta^+ & \cos\theta^+ \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

    To increase accuracy and to overcome distortions induced during finger image acquisition, the exact values of $\Delta_x^+, \Delta_y^+, \theta^+$ are added into A, and the average value is computed
    The implementation in Java allows Increment $(\Delta_x, \Delta_y, \Delta_\theta)$ in the accumulator array by adding evidence into accumulator $A$ of the corresponding bin
    $A_x[\Delta_x^+] = (A_x[\Delta_x^+]_N + 1).(A_x[\Delta_x^+]_{\Delta_x} + \Delta_x^+)/2$
    $A_y[\Delta_y^+] = (A_y[\Delta_y^+]_N + 1).(A_y[\Delta_y^+]_{\Delta_y} + \Delta_y^+)/2$
    $A_\theta[\theta^+] = (A_\theta[\theta^+]_N + 1).(A_\theta[\theta^+]_{\theta[\theta^+]} + \theta^+)/2$
Find indexes of maximum $A$, which are the most voted alignment parameters $[\Delta_x, \Delta_y, \Delta_\theta] = max(A)$

## 3.2. Discretized Rotation Based Approach

In the DRBA approach, it is common to consider all given points from *I* and *T* as possible corresponding points [11]. In addition, by checking if the direction difference of minutiae orientation is less than a defined threshold [12]. The second stage is to estimate *AP* from estimated corresponding pairs. The rotation angle is taken from the discretized data and used to compute *AP*. Translation parameters are computed using the affine transformation with the rotation angle from discretization data. At the third stage the accumulator array *A* is required to store all possible *AP*. The bin size is used to specify the step size in *A* and it is used when voting for the nearest bins of the current estimated *AP*. During the voting procedure it is general to cast the votes on the nearest bins, and the bin sizes are experimentally defined by considering different values from too small to large amounts [18]. The number of votes is accumulated by adding a vote for each computed parameters, shown in equation (1).

$$A[\Delta_x^+, \Delta_y^+, \theta^+] = A[\Delta_x^+, \Delta_y^+, \theta^+] + 1 \quad (1)$$

It is common in both approaches to define the accumulator array as a 3D array for rotation angles, and translations along the x and the y axis. The last step is to find the best alignment set, which can be one set or *N* sets of indexes of *A* with the largest votes. The implementation is shown in Algorithm 2.

**Algorithm 2:** An Alignment Algorithm for Discretized Rotation Based Approach

**Input:** Sets of minutiae points from $I$ and $T$.
**Output:** Set of $(\Delta_x, \Delta_y, \Delta_\theta)$.
Set direction tolerance $\theta_0$, and initialize Hough space into 3D array
Discretize values for $(\Delta_x, \Delta_y, \Delta_\theta)$ by initializing $A[\Delta_x, \Delta_y, \Delta_\theta]$
**for** *each minutiae* $m_t$ *in* $T$ **do**
  **for** *each minutiae* $m_i$ *in* $I$ **do**
    **for** *each rotation angle* $\theta^+$ *in* $\Delta_\theta$ **do**
      Compute direction difference $dd$ between $(\theta_i + \theta^+)$ and $(\theta_t)$
      dd = min(|$(\theta_i + \theta^+)$ - $\theta_t$|, 360 - |$(\theta_i + \theta^+)$ - $\theta_t$|)
      **if** $dd < \theta_0$ **then**
        Compute translation parameters

$$\begin{bmatrix} \Delta_x^+ \\ \Delta_y^+ \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \cos\theta^+ & -\sin\theta^+ \\ \sin\theta^+ & \cos\theta^+ \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

        To increase accuracy and to overcome distortions induced during finger image acquisition, $\Delta_x^+, \Delta_y^+, \theta^+$ are quantized to the nearest bins
        **for** $\Delta_\theta = (\theta^+ - Err_\theta) \rightarrow (\theta^+ + Err_\theta)$ **do**
          **for** $\Delta_x = (\Delta_x^+ - Err_\theta) \rightarrow (\Delta_x^+ + Err_x)$ **do**
            **for** $\Delta_y = (\Delta_y^+ - Err_y) \rightarrow (\Delta_y^+ + Err_y)$ **do**
              Increment $(\Delta_x, \Delta_y, \Delta_\theta)$ in the accumulator array by adding evidence into accumulator $A$
              $A[\Delta_x^+, \Delta_y^+, \theta^+] = A[\Delta_x^+, \Delta_y^+, \theta^+] + 1$

Find indexes of maximum $A$, which are the most voted alignment parameters $[\Delta_x, \Delta_y, \Delta_\theta] = max(A)$

**Pros**:

- Overcomes distortion when voting for nearest bins in all sets of alignment parameters.

**Cons**:

- General complexity of O(m*n*log(m*n)).
- Very inefficient for large number of minutiae points.
- Performs lots of operations.

### 3.3. Local Match Based Approach

In the LMBA approach, the first stage in this case is performed by using some methods to determine matching points, for example: by finding pair of points with similar Euclidean distance from their locations; or by first determining corresponding triangles between *I* and *T* [14] [15]; or by using similar triangles from Delaunay triangulation [15] , and then, estimate matching points from corresponding triangles. The second stage is performed by using the affine transformation with the computed rotation angle to compute *AP*. In the third stage it is common to define different bins of the accumulator array, e.g. starting from a large size of bins to the small size of bins to find the finer results of *AP*. The number of votes is accumulated by adding a number of aligned points determined after aligning points using each set of parameters, as shown in equation (2).

$$A[\Delta_x^+, \Delta_y^+, \theta^+] = A[\Delta_x^+, \Delta_y^+, \theta^+] + N \quad (2)$$

At the end of this approach, a set of *AP* with the highest number of votes is deemed as the one that represent the best transformation of tested sets of minutiae points [19]. The implementation and explanation of this algorithm is in Algorithm 3 and Algorithm 4.

**Algorithm 3:** An Alignment Algorithm for Local Match Based Approach

**Input:** Sets of minutiae points from $I$ and $T$.
**Output:** Set of $(\Delta_x, \Delta_y, \Delta_\theta)$.
Define $\theta_{length}$, $X_{length}$ ,$Y_{length}$ and number of matching levels e.g. $levels = 3$
Define required thresholds, $l_0$ for lengths of the triangles, and $a_0$ for the largest angle. Define possible alignment parameters for $(\Delta_x, \Delta_y, \Delta_\theta)$
Determine Delaunay triangles $DT_I$ and $DT_T$ for $I$ and $T$,respectively, and compute $invar\_features[]$
Compute alignment parameters as follows.
**if** $levels! = 0$ *and* $alignment\_score < set\_tolarance$ **then**
  Set the unit size for $A[\Delta_x, \Delta_y, \Delta_\theta]$
    $\Delta_{\theta_{size}} = \theta_{length}/2^{levels-1}$
    $\Delta_{x_{size}} = X_{length}/2^{levels-1}$
    $\Delta_{y_{size}} = Y_{length}/2^{levels-1}$
  Initialize $A[\Delta_{x_{size}}, \Delta_{y_{size}}, \Delta_{\theta_{size}}]$
  Set the bin size of rotation and translation
    $\Delta_{binsize} = 2^{levels-1}$
  Set $(\theta_{length}, X_{length}, Y_{length})$ = $(\Delta_{x_{size}}, \Delta_{y_{size}}, \Delta_{\theta_{size}})$
  **for** *each triangle in* $DT_T$ **do**
    **for** *each triangle in* $DT_I$ **do**
      Determine a triangle in $DT_I$ that correspond with a current triangle from $DT_T$ by For each corresponding triangle, compute alignment parameters using Algorithm 4
  Find index of maximum $A$ which is the most voted alignment parameters
  $[\Delta_x, \Delta_y, \Delta_\theta] = max(A)$

**Pros**:

- Alignment results are accurate.

**Cons**:

- Performs lots of computations.
- General complexity of O(m^2*n^2).
- Requires lots of multidimensional arrays.
- Very inefficient for large number of minutiae points.

## 4. COMPARATIVE ANALYSIS

Table I summarizes the performance and implementation of presented algorithms. Time complexity was calculated from the implementation of each algorithm. The LMBA shows a time complexity with high number of operations which leads to a low performance. This is because there are lots of computations involved when determining corresponding minutiae points from triangles. Another challenge is the memory required to process corresponding triangles which result in that the LMBA required more use of arrays and functions that are not supported in Java Card. In addition, in the LMBA, alignment process is performed for each computed set of alignment parameters. Therefore, computation time and operations are required to perform alignment.

The computational complexity of the DRBA is caused by involving the discretized rotations because this process requires repetitions of testing if the orientation differences for each pair of minutiae points are within the threshold. The AMPBA requires operations when computing the

average values of alignment parameters. However, the operations require most of supported instructions by the Java Card environment.

In Figure 1, the complexity in terms of instructions executed and operations by each algorithm is presented with respect to number of minutiae points or fingerprint features that need to be aligned.

Table I. Computational complexity of algorithms

| **Parameter** | **DRBA** | **LMBA** | **AMPBA** |
|---|---|---|---|
| Time Complexity | O(m*n*log(m*n)) | O(m^2*n^2) | O(m*n) |
| Space Complexity | O(n) | O(n) | O(1) |
| Use of Arrays | 60% | 90% | 40% |
| Unsupported Functions | 30% | 60% | 30% |

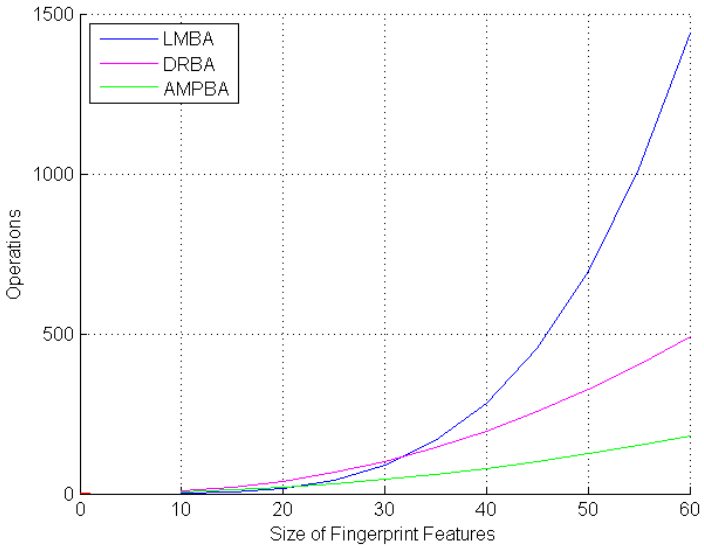


Figure 1. Operations performed with respect to fingerprint feature

## 5. CONCLUSIONS

From the above analysis it can be said that: the APMBA required less computational complexity compared to other algorithms although it increases as the number of fingerprint features increases but the time taken is less than that of the LMBA and DRBA. In addition, the implementation of the APMBA with one dimension arrays gives it advantages of simplified implementation. The LMBA requires more operations when the number of minutiae points increases, as a result is the slowest algorithm.

The future work is to study the implementation of promising matching algorithms on Java match on card to identify the one with least complexity in terms of time, memory and accuracy.

## ACKNOWLEDGEMENTS

The authors would like to acknowledge Department of Science and Technology, for funding this research.

## REFERENCES

[1] H. Pompi and D. A. Russell, (2012), Advances in fingerprint analysis. Angewandte Chemie International Edition 51, (15):3524–3531.
[2] V. Krithika and V. S. Kumar, (2011), Fingerprint identification: A brief literary review.
[3] J. Bringer H. Chabanne T. Chouta J. Danger M. Favre B. Mael, Y. Bocktaels and T. Graba, (2013), Studying potential side channel leakages on an embedded biometric comparison system. Database 4(5(7)).
[4] CardLogix Corporation, Smart Card Standards, (2010), http:// www.smartcardbasics.com/smart-card-standards.html, (Last visited 08/08/14).
[5] C. S. Mlambo, F.V. Nelwamondo, M.E. Mathekga, (2014), Comparison of effective Hough Transform-based fingerprint alignment approaches, International Symposium on Biometrics and Security Technologies, IEEE.(in press)
[6] ORACLE, "Java Card Technology Documentation" http://docs.oracle.com/javame/javacard/javacard.html, 2012. (Last visited 20/11/14).
[7] Precise Biometrics, (2013), "Match on Card", http://www.matchoncard.com/what-is-moc/smart-cards-and-fingerprint-recognition/, (Last visited 09/10/2014).
[8] ORACLE Inc. (2010), Java Card™ 3 Platform, Application Programming Notes.
[9] Infineon Ltd, (2014) "National ID", http://www.infineon.com/cms/en/product/smart-card-ic, 2014, (Last accessed 18/10/14).
[10] CardLogix Corporation, (2010) "Smart Card Standards", http:// www.smartcardbasics.com/smart-card-standards.html, 2010, (Last visited 08/10/14).
[11] A. Paulino, J. Feng and A. Jain, (2013), Latent Fingerprint Matching Using Descriptor-Based Hough Transform, IEEE Transactions on Information Forensics and Security, vol. 8, no. 1, pp. 31-45.
[12] R. Zhou, D. Zhong, and J. Han, (2013), Fingerprint Identification Using SIFT-Based Minutia Descriptors and Improved All Descriptor-Pair Matching, Sensors, ISSN: 1424-8220.
[13] F. Chen, X. Huang, and J. Zhou, (2013), Hierarchical Minutiae Matching for fingerprint and Palm print Identification, IEEE Trans- actions on Image Processing:a publication of the IEEE Signal Processing Society, vol. 22, no. 12, pp. 4964-497.
[14] G. Bebis, T. Deaconu, and M. Georgiopoulos. (1999), Fingerprint identification using Delaunay triangulation. Information Intelligence and Systems, 1999. Proceedings. 1999 International Conference on. IEEE, pp. 452–459.
[15] P. R. Mendes, A. C. Junior, and D. Menotti , (2010), A Complete System for Fingerprint Authentication using Delaunay Trian- gulation, Reconhecimento de Padroes, DECOM-UFOP, pp. 1-7.
[16] C.S. Mlambo, F.V. Nelwamondo, and M.E. Mathekga, (2014), "An improved Hough transform-based _fingerprint alignment approach", International Image Processing, Applications and Systems Conference, IPAS'14, IEEE,(Accepted.).
[17] C.S. Mlambo, M. Shabalala, M.E. Mathekga, and F.V. Nelwamondo, (2014), Application of Hough transform-based _fingerprint alignment on match on smart cards. International Conference on Cyber Warfare and Security ICCWS, (ICCWS-2015). Accepted.
[18] T. Uz, G. Bebis, A. Erol and S. Prabhakar, (2009), Minutiae-based Template Synthesis and Matching for Fingerprint Authentication, Computer Vision and Image Understanding, vol. 113(9), pp. 979-992.
[19] A. Gheibi and A. Mohades, (2013), Stable Geometric Fingerprint Matching, IET Computer Vision Journal.
[20] V. Gupta and R. Singh, (2012), Image processing and computer vision. Fingerprint Recognition CS676.
[21] A.C. Lomte, and S.B. Nikam, (2013), "Biometric fingerprint authentication by minutiae extraction using USB token system", International Journal Computer Technology and Applications, Vol. 4, No. 2, pp. 187-191.

[22] F. Benhammadi, and K. B. Beghdad, (2013), "Embedded Fingerprint Matching on Smart Card", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 27, No. 02.

## AUTHORS

**Cynthia S. Mlambo** is currently pursuing the Masters in Electrical Engineering at the University of Johannesburg. She holds an Honours Degree in Computer Engineering from the University of KwaZulu-Natal. Her areas of interest include image processing, pattern recognition and Smart ID Cards in biometrics.

**Meshack B. Shabalala** is a Biometric and Smart Card Researcher at the Council for Scientific and Industrial Research (CSIR), South Africa. He holds a Honours degree in Electrical Engineering from the University of the Witwatersrand.

**Fulufhelo V. Nelwamondo** is a Competency Area Manager for Information Security at the Council for Scientific and Industrial Research (CSIR), South Africa. He holds a PhD in Electrical Engineering from the University of the Witwatersrand and is a visiting professor of Electrical Engineering at the University of Johannesburg