Communication papers of the Federated Conference on Computer Science and Information Systems, pp. 193–197 ISSN 2300-5963 ACSIS, Vol. 13
DOI: 10.15439/2017F3

# Identification of Fingerprints using Circular String Approximation for Mobile Devices

Oluwole I. Ajala, Costas S. Iliopoulos, Mujibur R. Khan

Faculty of Natural
and Mathematical Sciences
Department of Informatics
King's College London
London, UK WC2R 2LS
Email:{oluwole.ajala, costas.iliopoulos, md.khan}@kcl.ac.uk

***Abstract*—Lately, Biometrics has employed the use of fingerprints to identify individuals. Out of the commonly used techniques, fingerprint authentication till date remains the most reliable. The bulk of research that has focused on fingerprint authentication, has however, neglected the issues that arise with fingerprints resulting to incorrect orientation identification. This is because it is assumed and often times wrongly, that the direction of the fingerprint will align with the stored fingerprint image. This singular issue poses tension in fingerprint matching, which only a negligible number of techniques have considered. As computers and mobile devices adopt fingerprint recognition as a way to authenticate the user, this apparent tension gains more popularity, becoming an integral research area which must be addressed.**

**Responding to this issue, this report proposes a novel pattern matching technique that caters for orientation differences in fingerprints, by implementing a pre-matching stage called the orientation identification stage and then match the fingerprint image with stored images using algorithms for approximate circular string matching. This fingerprint string information is now matched against a database of stored images using approximate string matching techniques.**

***Keywords*—Biometrics, Rotation,Fingerprint, Circular String Matching.**

## I. Introduction

The complication of security increases when information is disseminated over a wide area network or larger number of devices and systems being shared by unrelated users. As more information on individuals and companies are placed in the 'cloud', user privacy protection becomes a fundamental requirement. Issues on integrity, confidentiality and user authorisation pose a major concern. Thus, the core principles of information security are compromised.

Biometric resolutions proffer confidential and authentic transactions and also personal data privacy. Its usage transcends the commercial applications and law enforcement bodies to all government parastatals at all levels.

Fingerprints have provided an impeccable means of user authentication and personal identification for a long time, possibly dating back to the 19th century, when the records of fingerprint details of criminals in Argentina were released [20]. It has long since been adopted not just for law enforcement purposes (forensics and police) but also for commercial purposes like financial transactions and most recently, it is used as an authentication method in mobile devices and computers. Fingerprints are made up of minutiae, which are basically ridges and furrows in parallelism with each other. These minutiae form a complicated pattern that when impressed on a fingerprint scanner, leaves a print. These prints are matched to stored images on a database for either verification, authentication or both purposes [18].

## II. Road Map

The organisation of the rest of this paper is as follows. In section III, we present a very brief literature review, followed by section IV where we summarize our contribution to this problem. We present our approach in details in subsection IV-D. The experiment and the result analysis will be presented in section V. Finally, we briefly conclude and state the future work in section VI.

## III. Literature Review

As earlier mentioned, there have been extensive yet unsatisfactory algorithmic attempts at accurately processing and interpreting fingerprint patterns. Some of the hurdles encountered during the study of this overly researched area include but are not limited to application in comparing biological systems, signal processing, recovering information and correlating data from various files.

According to the Merriam-Webster dictionary, an algorithm is a procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation; a step-by-step procedure for solving a problem or accomplishing some end especially by a computer [15].

A string is processed by an algorithm to search for a text pattern within a larger body of text i.e. finding all the occurrences of a string/pattern (x) within a larger string (y.)

Generally, the algorithms probe the text through a 'window' starting by fastening at the left end of both text and window. They then compare the pattern to the text within the window space for either a match or a mismatch. This is called an attempt. They move to the next segment of the text and recheck for a match or mismatch. This process is repeated until

the whole text is read to the right end; a progression called sliding window mechanism [7].

Important algorithms in relation to this paper are the Needleman-Wunsch and Landau and Vishkin algorithms [14]. Approximate String Matching led to (i) Dynamic programming algorithm by Sellers and (ii) Landau and Vishkin algorithm [14]. The drawback of this algorithm was that it was expensive, space-consuming and inappropriate for long strings. Due to the complexities of matching strings across the grid and modification of edit distances, the exactness of the alignment within record time was challenging. Therefore, accuracy and speed became mitigating factors present in this algorithm.

Subsequently, a more operative algorithm that also took into consideration the no gap penalty was introduced individually by David Sankoff in 1972, T. K. Vintsyuk in 1968, Robert A. Wagner and Michael J. Fischer in 1974.

It was in an attempt to solve string matching more accurately in record time and use less space at a minimal cost that the Landau and Vishkin algorithm was presented [14]. The approach was to use suffix trees to reach more space over a minimal time frame. It takes into consideration that there could be errors or mismatches from unnecessary characters in both texts and patterns, insertions and deletions (k differences).

The algorithm had two stages: (i.) Pre-processing phase and (ii.) Iteration phase. Both text and pattern are processed in the pre-processing phase to construct a generalised suffix tree and to get the longest common extension (LCE). The suffix trees are used to compute the dynamic programming table in an accelerated time.

In the iteration phase, the algorithm probes each of the diagonals of the programming grid k times as far as it can go searching for all the pattern matches - permitting a minimum number of k differences.

'Diagonal' refers to all the cells with the same k differences. A simple stretch of a diagonal path defines an accurate match. [14]

## IV. Our Contribution

This paper focuses on the identification stage by proposing a fast novel pattern matching technique for fingerprint-based authentication on mobile devices that attempts to match fingerprints via classification. The fingerprint is intercepted with a series of scan circles and the minutiae information is derived. This information will then be translated into a string. This fingerprint string information is now matched against a database of stored images using an efficient, error tolerant, pattern matching algorithm. With this approach, identification of fingerprints can be done in linear time, with respect to the total length of all strings to be searched.

We devised a comprehensive method to identify fingerprints using the circular string matching technique. Because it exploits the power of circular string matching, the rotation of the input image is no longer a problem here. And the impact due to the distortion of the input image is kept at a minimal level by using approximation instead of exact matching. This is more like a proof of concept where we demonstrate that a comprehensive strategy could be devised by exploiting the power of circular string matching.

Few new things have been introduced that are used as underlying building blocks of our algorithm. They are presented in the following three subsections - IV-A, IV-B, and IV-C. Finally, the algorithm is presented in the subsection IV-D.

### A. Angle of Rotation

Let $x = x[0, 1, .., m-1]$ be any circular string of length m. We know, $x^i = x[i, i+1, .., m, 0, 1, ..i-1]$ is the $i$th rotation of $x$. The **angle of rotation** $a$ of $x^i$ in degrees is given by the following formulae

$$a = \frac{i}{m} \times 360^{\circ} \tag{1}$$

Hence,

$$i = \left\lceil \frac{a}{360} \times m \right\rceil \tag{2}$$

For example, if $x = ABCDEFGH$, then $x^1 = BCDEFGHA$ and $x^1$'s angle of rotation, $a$ is $45^{\circ}$. Similarly, if $y = 0123456789AB$ then the $45^{\circ}$ rotation of $y$ will be $y^2$, i.e. $23456789AB01$.

### B. Alignment of Concentric Circular Strings

Let us consider figure IV-B that shows the concentric circular strings of three different fingerprints.

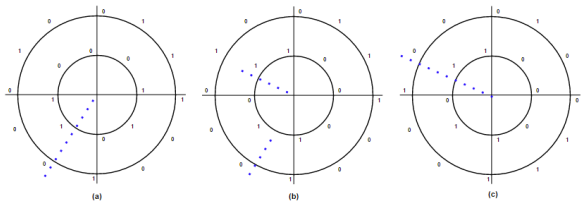

Fig. 1. Two concentric circles and the corresponding binary strings of three arbitrary images.

Only two circles are shown here for simplicity. The outer circle of figure IV-B(a) matches a rotation of the outer circle of figure IV-B(b) and the inner circle of figure IV-B(a) matches another rotation of the inner circle of figure IV-B(b). But the rotations of outer and inner circles are not aligned (as shown by the dotted lines). Hence, figure IV-B(b) is not a match of figure IV-B(a). On the other hand, the circular strings of figure IV-B(c) matches the ones in figure IV-B(a) and also the outer and inner circles are both having the same amount angle of rotation (say, $a^{\circ}$) i.e. they are aligned. Hence, figure IV-B(c) is the $a^{\circ}$th rotation of figure IV-B(a).

Therefore, in order to identify an input image as a rotation of a given fingerprint, all the circular strings of the input must match with a certain angular rotation of the corresponding circular string in the fingerprint.

In our algorithm, we take the longest circular strings from the input and the saved fingerprint. We check if any rotation of the circular string from the input matches the one from the saved fingerprint (or vice-versa) using **asmf/acdmf**. Suppose, the $a^{\circ}$th rotation of the input is found to be a good match. We then take the $a^{\circ}$th rotation of each of the remaining circular strings of the input image and match them with the

corresponding circular string from the saved image using the approximation algorithm of **Needleman-Wunsch**. We do not need to perform circular string matching for these strings.

Due to the rotation and distortion of the input image, it is less likely that the circular strings will match exactly. Hence, using the approximate matching is more likely to lead us towards a valid result.

### C. Novel Minutiae Extraction 2

Minutiae extractions are repeatedly used in our process. Hence, the speed of minutiae extraction has a significant impact on the overall speed of our process. That's why we have devised **Novel_Minutiae_Extraction2** witch is found to be extremely faster than the Novel_Minutiae_Extraction presented in Wole's MSc project paper [2]. Outline of Novel_Minutiae_Extraction2 is presented below:

```
ALGORITHM Novel_Minutae_Extraction2(char[][] img, int r, int cx, int cy)
// INPUT: "img", a 2d char array representing a fingerprint
// INPUT: "r", the radius of the circular string to be extracted
// INPUT: (cx, cy), the centre of the circular string to be extracted
// OUTPUT: "pattern", a circular binary string
{
      String mirrorredLower;
      String straightUpper;

      for (int i = (cx - r); i <= (cx + r); i++){
          double dJ = Math.Sqrt(Math.Pow(r, 2) - Math.Pow(i - cx, 2));
          if (dJ == 0.0)
            {
               if (i < cx)
                  {
                  if (pixel at img[i][cy] < 125) {
                        Append "0" to the end of straightUpper
                    } else {
                        Append "1" to the end of straightUpper
                    }
                  } else {
                    if (pixel at img[i][cy] < 125) {
                        Append "0" to the start of mirrorredLower
                    } else {
                        Append "1" to the start of mirrorredLower
                    }
                  }
              } else if (dJ == (int)dJ) {
                  if (pixel at img[i][ cy + (int)dJ] < 125) {
                     Append "0" to the end of straightUpper
                    } else {
                     Append "1" to the end of straightUpper
                    }
                  if (pixel at img[i][ cy - (int)dJ] < 125) {
                     Append "0" to the start of mirrorredLower
                    } else {
                     Append "1" to the start of mirrorredLower
                    }
              }
        }

        return mirrorredLower + straightUpper;
}
```

### D. Outline of the Algorithm

We have outlined the complete process of Fingerprint matching as below:

1) Construct the database of circular strings taking the centre of the image as the centre. Draw concentric circles and then convert them into strings of 0's and 1's (using the Novel_Minitiae_Extraction2 as outlined in IV-C). Say $X_{ir}$ be the circular string for the $i$th image with a radius $r$.
2) Save all rotations of each of the circular strings in the database. Let $DR_{ir}$ be the collection of rotated strings from $X_{ir}$ e.g. if $X_{ir}$ = "1011" then $DR_{ir}$ will be "1011","0111","1110","1101"
3) For the input fingerprint, we shall identify the boundary of the input. Let $Gv$ be the larger among the stretches in the height and width of the image. (See figure 3)

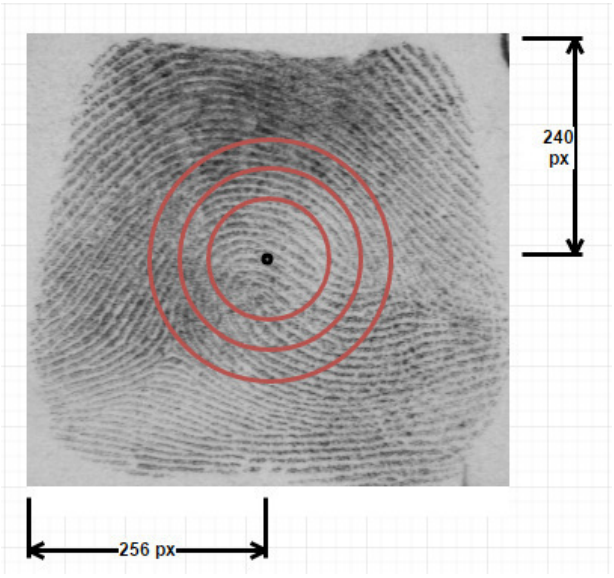


Fig. 2. A sample image from the database. (H:480px, W:512px)

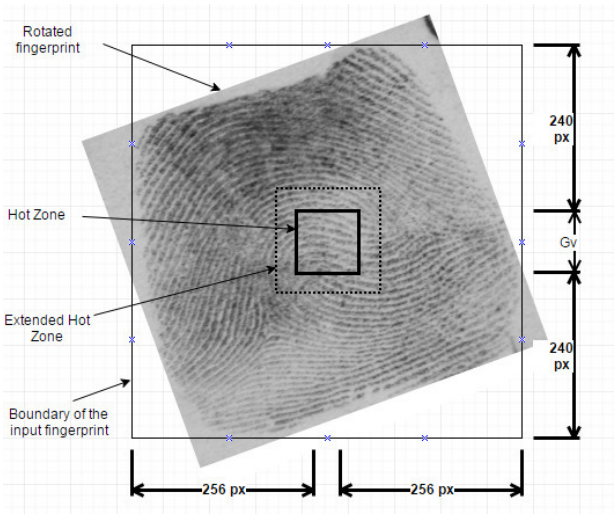


Fig. 3. Image boundary of a rotated input image

4) Let's draw a square with sides of length $Gv$ at the centre of the boundary, as shown in the image (alternatively, we could draw a circle with radius $Gv$ from the centre of the identified boundary). It is most likely that the centre of the original image lies somewhere within this hot zone.
5) We could add an arbitrary amount of tolerance ($dT$) to $Gv$ and get the extended hot zone. The centre of the original image is further likely to fall somewhere within the extended hot zone.
6) Now, start at the centre of the identified boundary and

take it as the prospective centre (in the subsequent iterations, we shall spiral outwards from this point until the boundary of the extended hot zone). (See figure 4)

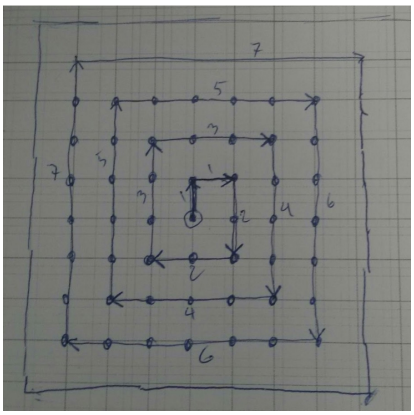

Fig. 4. Spiral outward from the centre

7) Draw concentric circles of various radius from this prospective centre. Then construct circular binary strings for each of the circle (using the **Novel_Minitiae_Extraction2** as outlined in subsection IV-C). Make a sorted list **CsR** of circular string and radius pairs, $< CStr, Radius >$, in descending order of the length of **CStr**.
8) a) Apply **asmf/acdmf** method to compare all the rotations of **CsR[0].CStr** (i.e. the first circular strings obtained in step 7) with the one in $DR_{ir}$ where $r = CsR[0].Radius$. Let's assume the $q$th rotation of $CsR[0].CStr$ has the highest match. Calculate the angle a for the$q$th rotation using eq. 1.
   b) For each **CsR[p].CStr** where $1 \leq p < len(CsR)$, find the $q$th rotation of **CsR[p].CStr**, where $q$ is calculated for angle a using eq. 2. Then apply **Needleman-Wunsch** method to compare the $q$th rotations of **CsR[p].CStr** with the one in $DR_{ir}$ where $r = CsR[p].Radius$.
9) Find the average of matching obtained in steps 8a and 8b.
10) a) If the percentage of matched strings is higher than a given threshold (**tS**), then save the sum of percentage of match and mark the $i$th image as a candidate and select the next ( $(i+1)$th) image. Proceed to step 11.
    b) Otherwise, repeat from step 6 by choosing a different centre point.
    c) Repeat it until all the points within the extended hot zone are tried.
11) Repeat step 6 to step 10 until all the images in the database are exhausted.
12) Return the top ranked images.

## V. Experimental Results

We coded the program in C# using Microsoft Visual Studio 2013. Intel Xeon processor with 3.5 GHz, the Operating system was 64 bit Windows 7.

Figure V is an input panel with H:1024 and W:2048 pixels that simulates the input screen of an arbitrary mobile device. The original fingerprint is a $512 \times 512$ pixel bitmap. Which is rotated and placed at a random location inside the input panel. Then the boundary of the input image is identified.

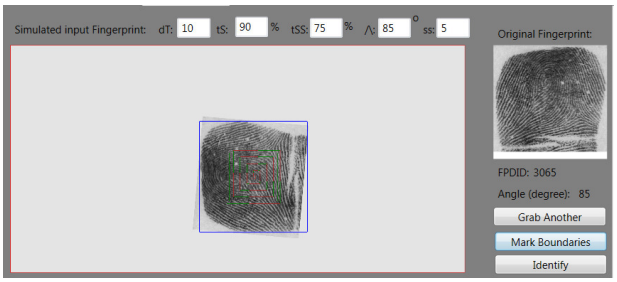


Fig. 5. Input panel with a simulated input and its identified image-boundary.

The performance and accuracy of the proposed algorithm vastly depends on the size of the database as well as on the number of probable centre points on the input image. This number varies depending on the rotation of the input image. We have collected data for this against the degree of rotation which is presented in Table I.

TABLE I: The average number of points considered as potential centre of the rotated image (rounded to nearest 100), against the angle of rotation of the input image.

| **Angle of rotation of the input image (in degree)** | **Average number of prospective centres (dT=10)** |
|---|---|
| 0, 90, 180, 270 | 1500 |
| 15, 105, 195, 285 | 15000 |
| 30, 120, 210, 300 | 30000 |
| 45, 135, 225, 315 | 40000 |
| 60, 150, 240, 330 | 30000 |
| 75, 165, 255, 345 | 15000 |

Table II shows the result against three different database sizes. For every input, we captured the top 5 ranked images from the database. The second column shows the percentage of times when the correct fingerprint is returned as the top ranked image. The third column shows the percentage of times when the correct image is not the top ranked but is withing the top 5.

TABLE II: Summery of findings (dT=10, tS=90, tSS=75).

| **Db Size** | **% of cases where the correct fingerprint is top ranked** | **% of cases where the correct fingerprint is in top 5** | **% of cases where the correct fingerprint is Not in top 5** | **Average time (Seconds)** |
|---|---|---|---|---|
| 50 | 72.72% | 09.09% | 18.18% | 881.30 |
| 100 | 47.62% | 23.81% | 28.57% | 3276.93 |
| 500 | 14.29% | 28.57% | 57.14% | 63644.37 |

It is clearly visible that the performance degrades significantly as the database size increases. That indicates the huge potential for the future scope to improve the algorithm.

## VI. Conclusion

This report establishes yet another example of an interdisciplinary effort where mathematical algorithms have been translated into codes in computer science that are used to solve problems extending beyond bioinformatics.

As more systems and devices adopt fingerprint recognition as biometric authentication, the need for effective pattern matching cannot be over emphasised. To this end, a plethora of algorithms and string matching techniques have erupted. This paper addressed the ever rising issue with fingerprint authentication- orientation. With the correct orientation figured out, matching becomes much easier and faster thus speed and accuracy is maximised.

## References


[1] Ajala, O., Aljamea, M., Alzamel, M., Iliopoulos, C. S., Fast Fingerprint Rotation Recognition Technique Using Circular Strings in Lexicographical Order. SAI Intelligent Systems Conference 2016, September 21-22, 2016, London, UK

[2] Ajala, O., Aljamea, M., Alzamel, M., Iliopoulos, C. S., Strigini, Y., Fast Fingerprint Recognition Using Circular String Pattern Matching Techniques. PATTERNS 2016 : The Eighth International Conferences on Pervasive Patterns and Applications

[3] Al-Jamea, M et al, 2015. A Novel Pattern Matching Approach for Fingerprint-based Authentication. PATTERNS 2015: The Seventh International Conferences on Pervasive Patterns and Applications, [Online]. 978-1-61208-393-3, 45-49. Available at: https://www.thinkmind.org/download.php?articleid=patterns_2015_2_40 70032 [Accessed 12 March 2016].

[4] Angle, S et al, 2005. Biometrics: A Further Echelon of Security. The Impact of Technology on Plagiarism Prevention and Detection: Research Process Automation, a New Approach for Preven tion, 3, 3-9.

[5] Barnes, J. G, 2014. The Fingerprint Sourcebook. 1st ed. UK: Create Space Independent Publishing Platform.

[6] Barton, Iliopoulos, Pissis, C., C. S., S. P., 2014. Fast algorithms for approximate circular string matching. Algorithms for Molecular Biology, 9:9, 10.

[7] Christian, C., Lecroq, T., Handbook of exact string matching algorithms. London, UK:: Kings College Publications, 2004

[8] Colussi, L., 1991. Correctness and efficiency of pattern matching algorithms. Information and Computation, 95, 225251. https://doi.org/10.1016/0890-5401(91)90046-5

[9] Cormen, Leiserson and Rivest, T. H., C. E., and R. L. , 1990. Introduction to Algorithms. 1st ed. Cambridge, Massachusetts: MIT Press.

[10] Dongjae, D.L, 2008. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on Year: 2008, Volume: 38, Issue: 1. Recognizable-Image Selection for Fingerprint Recognition With a Mobile-Device Camera, 38, Issue 1, 233 243.

[11] Donida, L.R, 2013. Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2013 IEEE International Conference on Year: 2013. Accurate 3D fingerprint virtual environment for biometric technology evaluations and experiment design, DOI: 10.1109/CIVEMSA.2013.6617393, 43-48.

[12] Galton, F., 1892. Finger Prints. 1st ed. London, New York: Macmillan and Co. http://www.explainthatstuff.com/. 2004. Biometric fingerprint scanners. [ONLINE] Available at: http://www.explainthatstuff.com/fingerprintscanners.html. [Accessed 31 August 16].

[13] Gao, Q, 2014. A preliminary study of fake fingerprints. I.J Computer Network and Information Security, 12, 1-8.

[14] Landau, Vishkin, G.M., U., 1998. Fast String Matching with k Differences* .Journal of Computer and System Sciences, 37, 63-78. https://doi.org/10.1016/0022-0000(88)90045-1

[15] Merriam-Webster. 2016. Algorithm. [ONLINE] Available at: http://www.merriam-webster.com/dictionary/algorithm. [Accessed 09 September 16].

[16] Miranda, R. C., Ayala-Rinc on, M., Solon, L, 2005. Advances in Bioinformatics and Computational Biology.. 1st ed. Sao Leopoldo, Brazil: Springer Berlin Heidelberg.

[17] Moses, K.R, 2014. The finger print source book. 1st ed. UK: CreateSpace Independent Publishing Platform

[18] Sebastian, S., Literature survey on automated person identification techniques, International Journal of Computer Science and Mobile Computing, vol. 2, no. 5, pp. 232237, 2013

[19] Sheik, S. S., Aggarwal, S. K., Poddar, A., Balakrishnan, N., Sekar, K., 2004. A FAST Pattern Matching Algorithm. J. Chem. Inf. Comput. Sci, 44, 1251-1256. https://doi.org/10.1021/ci030463z

[20] Unar, J., Seng, W. C., and Abbasi, A., A review of biometric technology along with trends and prospects, Pattern Recognition, vol. 47, no. 8, pp. 26732688, 2014 https://doi.org/10.1016/j.patcog.2014.01.016