Heap-based Algorithms to Accelerate Fingerprint Matching on Parallel Platforms

Ricardo J. Barrientos[1(✉)][0000−0001−5345−7061], Ruber Hernández-García[1][0000−0002−9311−1193], Kevin Ortega[2], Emilio Luque[3][0000−0002−2884−3232], and Daniel Peralta[4,5][0000−0002−7544−8411]

[1] Laboratory of Technological Research in Pattern Recognition (LITRP), Department of Computer Science and Industries, Faculty of Engineering Science, Universidad Católica del Maule, Talca, Chile
{rbarrientos, rhernandez}@ucm.cl
[2] Kunert Business Software GmbH (KBS-Leipzig), Leipzig, Germany
kevin.ortega@kbs-leipzig.de
[3] Universitat Autònoma de Barcelona, Barcelona, Spain
emilio.luque@uab.es
[4] Data Mining and Modelling for Biomedicine Group, VIB Center for Inflammation Research, Ghent, Belgium
daniel.peralta@irc.vib-ugent.be
[5] Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

Abstract. Nowadays, fingerprint is the most used biometric trait for individuals identification. In this area, the state-of-the-art algorithms are very accurate, but when the database contains millions of identities, an acceleration of the algorithm is required. From these algorithms, Minutia Cylinder-Code (MCC) stands out for its good results in terms of accuracy, however its efficiency in computational time is not high. In this work, we propose to use two different parallel platforms to accelerate fingerprint matching process by using MCC: (1) a multi-core server, and (2) a Xeon Phi coprocessor. Our proposal is based on heaps as auxiliary structure to process the global similarity of MCC. As heap-based algorithms are exhaustive (all the elements are accessed), we also explored the use an indexing algorithm to avoid comparing the query against all the fingerprints of the database. Experimental results show an improvement up to 97.15x of speed-up, which is competitive compared to other state-of-the-art algorithms in GPU and FPGA. To the best of our knowledge, this is the first work for fingerprint identification using a Xeon Phi coprocessor.

Keywords: Coprocessors, Xeon Phi, MCC, Fingerprint

1 Introduction

Fingerprint identification is the most used biometric method to automatically recognize the identity of a person [17], thanks to its usability and reliability

[23]. Fingerprints are the most studied biometric trait [11], and different algorithms have been proposed since 1975 to deal with them in their acquisition [20], processing [5], classification [11], and matching [6].

A fingerprint consists of a set of curves or lines, known as ridges. A ridge is defined as a single curved segment, and a valley is the region between two adjacent ridges. The discontinuities in the ridges, such as terminations and bifurcations, are called minutiae (see Figure 1(a)). Formally, minutiae are points typically represented as a triplet (x, y, θ), where x and y represent the point coordinates and θ is the ridge direction at that point. These unique features are mathematically represented as a biometric template (also called template), which is stored in a biometric database. These templates are used in different ways for matching purposes. Although there are several fingerprint matching algorithms, the most common approaches are based on minutiae [11]. The objective of a minutiae-based algorithm is to find the maximum quantity of matching between pairs of two fingerprints.

Minutia Cylinder-Code (MCC) [6] is an accurate algorithm for fingerprint identification, which takes 45 milliseconds for one comparison between two fingerprints. Thus, it implies 45 seconds for a database with 1000 fingerprints, which is a considerable time, especially when the database reaches the order of tens thousands or more fingerprints. There are two usual methods to decrease the execution time in this case, which are (1) to avoid comparing all fingerprint pairs by using classification methods [8, 16] or indexing algorithms [7, 4], and (2) to accelerate the matching processing by using parallel computing [27, 21].

In this work, we explored to use both methods aiming to accelerate fingerprint matching process by using MCC. For this purpose, we employ a Xeon Phi coprocessor, which is one of the most promising alternatives for algorithms acceleration in the current technological context [29]. Besides, our proposal is based on heaps as auxiliary structure to process the global similarity of MCC. As heap-based algorithms are exhaustive (all the elements are accessed), we also use an indexing algorithm to avoid comparing the query against all the fingerprints of the database. There are several indexing algorithms to accelerate searching process on metric spaces [9, 26, 10]. In our approach, we selected the List of Clusters index [9] because it has shown good properties in different parallel platforms previously [25, 12, 1].

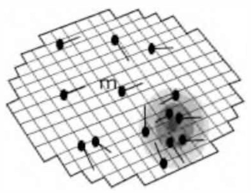
Thus, we propose a parallel algorithm based on heaps using a Xeon Phi coprocessor with the aim to obtain a high speed-up over the sequential counterpart, and showing how suitable is this parallel platform for fingerprint identification. To the best of our knowledge, this is the first approach with this coprocessor to accelerate fingerprint matching.

2 Related Work

2.1 Minutia Cylinder-Code

The Minutia Cylinder-Code algorithm [6], is currently one of the best methods to represent fingerprints and to execute a match making process. The main idea of

(a) Bifurcations and ridge endings are depicted in blue and red color, respectively. The orange circle locates the core of the fingerprint.



(b) Example of a neighborhood $N_{p^m_{i,j}}$: A cylinder section representing a neighborhood $N_{p^m_{i,j}}$ that is associated to a given minutia m in the center of this cylinder section. The darker section is the place where the highest point $p^m_{i,j}$ value lays (center of a cell), therefore it represents a higher contribution value C_m. The minutiae that are within the dark zone, are the neighborhood $N_{p^m_{i,j}}$.

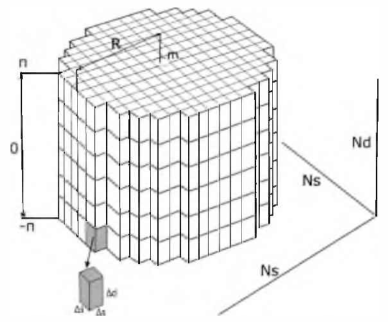
Fig. 1. Minutiae and their representation by MCC.

the representation of the MCC is to generate a local structure for each minutia m in a given template T, where the structure is created by a spatial and directional relationship between the minutiae and their neighborhood, which is set according to a fixed radio [14]. Each local point is associated with a 3D structure, called cylinder. This cylinder is associated with each minutia m of a fingerprint (see Figure 1(b)).

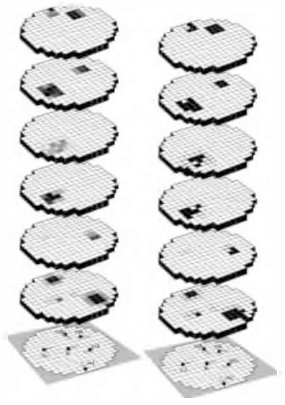
Figure 2(a) depicts the local structure of a cylinder. Each cylinder is centered on a minutia m that has a fixed radius and a height (from $-\pi$ to π) of 2π. In addition, all cylinders have the same size, because the fixed ratio. Each cylinder is discretized in $N_S \times N_S \times N_D$ cells. Each cell is a small cuboid with $\triangle_S \times \triangle_S$ base and $\triangle_D$ height, where $\triangle_S = \frac{2R}{N_S}$ and $\triangle_D = \frac{2\pi}{N_D}$ [6]. N_S is defined as the 2D space around a minutia m $N_S \times N_S$ and N_D represents the number of divisions that are applied to the height of 2π which represents the angular distance [14].

A numerical value is associated to each cell, also known as contribution $C_m(i, j, k)$, which is the sum of the contributions of each minutia mt belonging to the neighborhood $N_{p^m_{i,j}}$ at the point $p^m_{i,j}$ [22]. Based on the location and direction of each minutia in the neighborhood $N_{p^m_{i,j}}$, the values of both spatial and directional contributions are higher when: (1) the location is close to the point $p^m_{i,j}$, and (2) the direction is close to the angle set by $d\varphi_k$. By way of explanation [7], the value of a cell is a probability. This probability is higher when a minutia is close to a cell and its direction is similar to the value $d\varphi_k$ (see Figure 2(b)).

A characteristic of MCC is that it can represent the value of each cell as a bit [7, 6]. In this work, we propose algorithms to perform the matching of fingerprints through the binary cylinders obtained from the MCC. The local similarity between two cylinders can be simply estimated by applying simple



(a) Local structure of a cylinder associated to a minutia m in MCC.



(b) Real (left) and binary (right) representation of the same cylinder.

Fig. 2. Local structure of a cylinder and its graphical representations.

bitwise operations (*and, exclusive-or, population-count*) between the two corresponding binary vectors[6]. Thus, in order to compare two fingerprints, an overall score denoting their global similarity, has to be obtained from the local similarities. We used in the experiments as global similarity the Local Similarity Sort (LSS). This technique sorts the local similarity values and computes the average of the best n values, where the value of n is defined according to [6] by $n = min_{np} + \lfloor (Z(min\{n_A, n_B\}, \mu_P, \tau_P)) \cdot (max_{np} - min_{np}) \rceil$, where n_A and n_B are the number of minutiae of the templates to be compared; $\lfloor \cdot \rceil$ denotes the rounding operator; and Z is the function defined as $Z(v, \mu, \tau) = \frac{1}{1+e^{-\tau(v-\mu)}}$. We used the same values proposed in [6] for $\mu_P = 20$, $\tau_P = 2/5$, $min_{np} = 4$, and $max_{np} = 12$.

2.2 Intel Xeon Phi coprocessor

The Intel Xeon Phi coprocessor [28, 29] consists of 61 to 72 cores connected by a high performance on-die bidirectional interconnect. The coprocessor runs a Linux operating system and supports all main Intel development tools like C/C++, Fortran, MPI and OpenMP. The coprocessor is connected to an Intel Xeon processor (the host) via the PCI Express (PICe) bus. It is mainly composed of cores, which have one Vector Processing Unit (VPU), and one L1 and L2 cache per core. In the VPU each operation can be a fused multiply-add giving 32 single-precision or 16 double-precision floating-point operations per cycle. This architecture has a complex vector unit. However, empirical studies show that the efficient exploitation of the vector unit is crucial to achieve a significant performance improvement [28]. A limitation of using a Xeon Phi, such as in other coprocessors, is to deal with the transfers to/from the CPU.

2.3 Approaches based on Parallel Computing

The algorithm MCC has shown very high performance in terms of accuracy, but it requires a high computation cost. In the Parallel Computing area, the use of coprocessors to accelerate processing is being exploited, and currently there are three main coprocessors used for this purpose: GPU, FPGA, and Intel Xeon Phi.

In the present work, beside proposing and implementing a multi-core algorithm, we also developed a Xeon Phi algorithm, which is to the best of our knowledge the first algorithm with this coprocessor to accelerate fingerprint matching. We compare our algorithms to previous state-of-the-art approaches that use different parallel platforms with the MCC algorithm [15, 18, 22].

Gutierrez et al. [15] implement an algorithm based on GPU for the MCC. They use two different models of GPU in their experiments. They implement an exhaustive algorithm where all the fingerprints are accessed by each query. They achieve a speed-up of 29.0x with the LSS similarity and $N_s = 8$. Lindoso et al. [22] propose an algorithm based on a FPGA for the fingerprint matching, achieving a speed-up of 23.7x. Other work based on FPGA is proposed by Jiang and Crookes [18], which is to the best of our knowledge the FPGA-based algorithm with highest performance, achieving a speed-up of 46.5x.

3 Parallel Computing Algorithms

In this section we show our proposed algorithms for fingerprint identification on a multi-core server (Section 3.1) and a Xeon Phi coprocessor (Section 3.2). We also explored the use of an indexing method to accelerate the search by discarding elements with a pre-processing of the database (Section 4.1).

3.1 Multi-core Algorithm

We take as input parameter the cylinders of each fingerprint, where each cylinder is an array of bits. This algorithm is illustrated in Figure 3, where we distributed the queries among the threads of the multi-core server. Each thread performs the similarity function against all the elements of the database according to the similarity of cylinders described in Section 2.1. For avoiding O.S. resource conflicts, we execute each thread in an exclusive core.

In our algorithm the data (database and queries) is store in two matrices, but for this we keep an index indicating where start and end the cylinder of each fingerprint. Thus, we used a heap [19] as auxiliary structure to keep the n_p highest similarities among cylinders. Each thread creates its own heap as a private variable. This is required for the global similarity LSS that we used as the global and final similarity.

3.2 Xeon Phi Algorithm

Given a fingerprint database of N_{DB} fingeprints. Let $Fingeprints$ be the matrix storing the cylinders of N_{DB} fingeprints in the database; $Queries$ be the matrix storing $num\_queries$ queries; and ID_thread be the current thread ID and

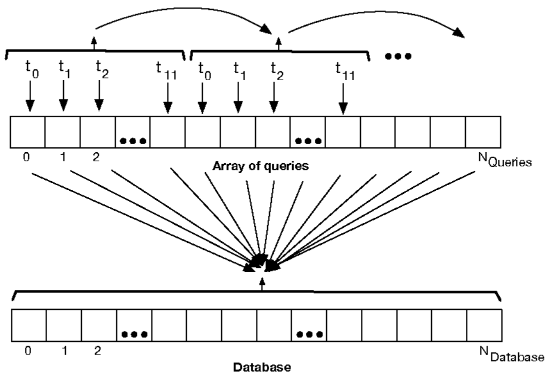


Fig. 3. Illustration of the muti-core algorithm.

num_threads the total number of threads. We propose an algorithm based on MCC for a Xeon Phi coprocessor, which is presented in Algorithm 1. The queries are accessed in a round robin manner (line 4 in Algorithm 1) by the thread of the Xeon Phi. Each thread calculates a local similarity between its query and the fingerprints in the database (line 14), and then the global similarity (line 26) with the values stored in its heap of size n, which is the overall similarity score between the two fingerprint templates to be compared.

This algorithm does not require any synchronization instructions. The only synchronization of the algorithm is carried out implicitly when the pragma offload region ends. It is noteworthy that this algorithm has focused on processing the queries in batches of size *num_queries*.

4 Experimental Results

In our experiments, we used a multi-core server composed of two Haswell architecture Intel Xeon E5-2620v3 2.4GHz, i.e. 12 hyperthreading cores with 32GB in main memory. The coprocessor used in the experiments was an Intel Xeon Phi 7120 with 61 cores, supporting up to 4 threads per core, and 16GB of memory. We used all cores in our Xeon Phi algorithm, and 4 threads per core. More details are shown in Table 1.

Table 1. Platforms description.

(a) Intel Xeon Phi details.

Coprocessor	Intel Xeon Phi 7120P
Cores	61 cores of 1.24 GHz / 4 threads per core
Memory	16GB of memory (bandwidth 352 GB/s)
Cache L1	3.8MB L1 (64KB L1 per core)
Cache L2	30.5MB L2 (512KB L2 per core)
Compiler	icc version 15.0.2, flags: -03 -openmp

(b) Platform for sequential and multi-core versions

Processor	2xIntel Xeon E5-2660v3, 12 cores with HT 15MB Cache, 2.40 GHz, Haswell
Memory	32 GB, 59.7 GB/s
Operative System	GNU Debian System Linux kernel 3.10.0-229.4.2.el7.x86_64
Compiler	icc version 15.0.2, flags: -03

Algorithm 1 Algorithm in Xeon Phi based on the MCC.

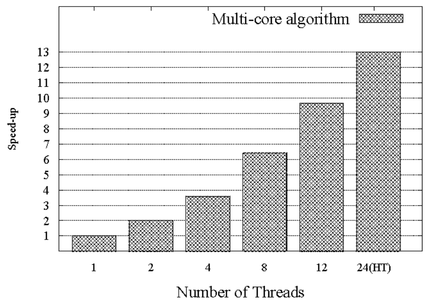
```
#pragma offload in(Fingeprints) in(Queries) in(Heaps)
{
  #pragma omp parallel
  for ($i = ID\_thread$; $i < num_{queries}$; $i$ += $num\_threads$)
    for ($j = 0$; $j < N_{DB}$; $j$++)
    {
      first_cyl = start_cylinders($Fingeprints$, j);
      size_cyl = size_cylinders($Fingerprints$, j);
      first_cyl_Q = start_cylinders_query[i];
       size_cyl_Q = size_cylinders_query[i];
       for ($k$=first_cyl_Q; $k$ <first_cyl_Q+size_cyl_Q; $k$++)
         for ($l$=first_cyl; $l$ <first_cyl+size_cyl; $l$++)
         {
           similarity = $1 - \frac{\|Fingerprints[l] XOR Queries[k]\|}{\|Fingerprints[l]\| + \|Queries[k]\|}$
           if (Heap[$ID\_thread$].size() $>=$ $n$)
           {
             if (Heap[$ID\_thread$].top() $<$ similarity)
             {
               Heap[$ID\_thread$].pop();
               Heap[$ID\_thread$].push(similarity);
             }
             else
               Heap[$ID\_thread$].push(similarity);
           }
         }
        global_similarity = $LSS$(Heap[$ID\_thread$]);
    }
}
```

We used the NIST (National Institute of Standards and Technology) [30] as database, which contains 27,000 pairs of segmented 8-bit gray scale fingerprint images. We used the first 27,000 impressions as database and the 27,000 second impressions as queries. The average number of minutiae is 206.9 with a maximum of 610.

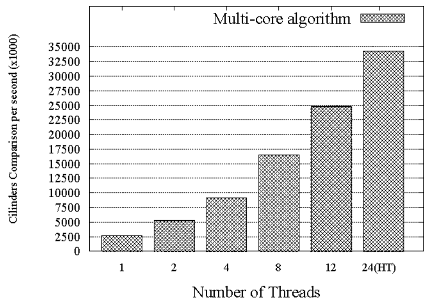
Figure 4 shows performance measures between the sequential and multi-core method. Figure 4(a) shows the speed-up of the multi-core algorithm, and Figure 4(b) shows the quantity of cylinders comparison operations performed per second. Both experiments in Figure 4 execute one thread in one exclusive core, and for the label 24(HT) we executed two threads per core using the hyperthreading property of the processors. We observe a good scaling behavior when a thread is executed in an exclusive core, but also with 24 threads (taking into account the shared resources for hyperthreading cores).

Figure 5 shows the speed-up of the sequential and multi-core algorithms with 12 and 24 HT threads against the algorithm in Xeon Phi. It should be highlighted each core in Xeon Phi supports up to 4 threads executed simultaneously. Our algorithm in Xeon Phi reaches 97.51x of speed-up because of the good use of cache when threads executed in the same core access to the same data ([28]). This behavior occurs when threads processing different queries must access to the same elements of the database.

There are different previous works covering the fingerprint identification in the coprocessors GPU and FPGA, which were described in Section 2.3. We compare our results to these previous state-of-the-art algorithms in the Table



(a) Speed-up of the multi-core algorithm



(b) Quantity of Cilinders comparison per second

Fig. 4. Performance measures between the Sequential and multi-core version. HT=Hyperthreading.

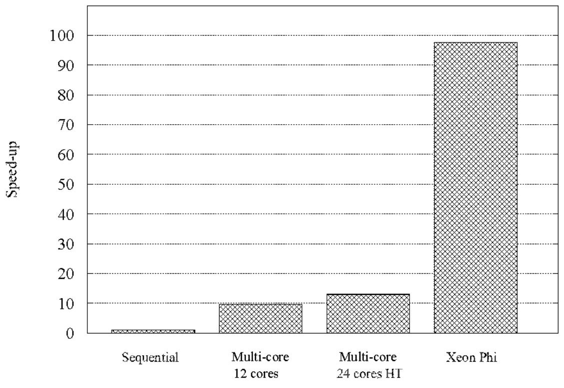


Fig. 5. Speed-up between of the multi-core and Xeon Phi version. Values obtained over the sequential algorithm. HT=Hyperthreading.

2. It should be noticed that FPG-based approaches use different databases and they achieve a speed-up up to 46.5x. Our algorithm in Xeon Phi achieves a competitive speed-up, showing that a Xeon Phi coprocessor can be used for fingerprint matching and specifically for the MCC algorithm.

4.1 Indexing Algorithm for MCC

In recent years, in the area of similarity search, many indexing approaches have been proposed ([10]). The objective of these indexes is to avoid distance calculations between the query and each of the elements of the database. This is performed by discarding elements using the triangle inequality property of a metric space. In this work, we implemented the List of Cluster (LC) [9] index

Table 2. Comparison with state-of-the-art algorithms on different parallel platforms.

Algorithm	**Parallel Platform**	**Database**	**Speed-up**
Multi-core algorithm	24 (HT) threads	NIST Special Database 14	12.99x
Xeon Phi algorithm	Xeon Phi 7120	NIST Special Database 14	97.51x
GPU algorithm with LSS in [15]	NVIDIA GeForce GTX 680	NIST Special Database 14	29.0x
GPU algorithm with LSS in [15]	NVIDIA Tesla M2090	NIST Special Database 14	24.9x
FPGA method in [18]	FPGA Xilinx Virtex-E	10,000 fingerprints randomly generated	46.5x
FPGA method in [22]	FPGA Xilinx Virtex-4 LX	56 fingerprints	23.7x

to be used with the MCC algorithm, and trying of discarding fingerprints of the database.

We used an adaptation of the cylinder similarity function to be able of using the LC index, which requires that the data (X) and similarity function (d) be a metric space [31]. This means that d must hold the following properties on X: (1) strict positiveness ($d(x, y) > 0$ and if $d(x, y) = 0$ then $x = y$), (2) symmetry ($d(x, y) = d(y, x)$), and the triangle inequality ($d(x, z) \leq d(x, y) + d(y, z)$).

We used the similarity function shown in [7], where given two fingerprints F_1, F_2, and S_1 S_2 be the corresponding sets of n-dimensional binary vectors, a similarity measure between F_1 and F_2 can be defined as follows:

$$sim(F_1, F_2) = \frac{\sum_{s \in S_1} max_{s_j \in S_2}\{H(s, s_j)\}}{|S_1|} \tag{1}$$

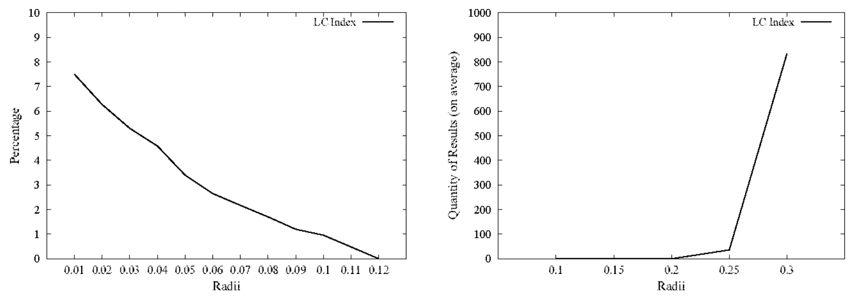
where $|S_1|$ is the cardinality of set S_1, and H is a normalized similarity measure between two binary vectors, based on the Hamming distance (d_H), defined as follows:

$$H(a, b) = \left(1 - \frac{d_H(a, b)}{dim}\right) \tag{2}$$

where dim is the dimension of the binary vectors, and p is a parameter controlling the shape of the similarity function (we set $p = 30$ for the experiments).

We selected the LC index because: (1) it has been previously used with parallel platforms ([2, 24, 13]) (2) they hold their indexes in dense matrices which are very convenient data structures for mapping algorithms onto Xeon Phi. This index can be implemented dividing the space in two different ways: taking a fixed radius for each partition or using a fixed size. To ensure good load balance in a parallel platform, we consider partitions with a fixed size of K elements, thus the cluster radius rc with the center C is the maximum distance between its center and its K^{th} nearest element.

We did not achieve positive results using the LC index, obtaining execution times higher than the exhaustive approaches. This is because this index was not able to discard elements with the elements of the MCC algorithm. We can



(a) Percentage of Discarded Clusters (b) Quantity of Retrieved Elements

Fig. 6. Average values using the List of Cluster index.

see in Figure 6(a) the percentage (on average) of discarded elements when the radius of the query is increased, and Figure 6(b) shows the number of retrieved elements when the radius of the query is increased. We observe that when the radius is large enough to retrieve elements, the LC is not able to discard any elements. This phenomenon has been observed before, and it is named as *curse of dimensionality* ([3]), where the intrinsic dimensionality ([10]) is high enough to avoid discarding, and the distance histogram of the elements is more concentrated when the dimension grows. Despite the fact that we did not achieve a good result in performance, we decided to add this section to conclude that a metric index like the LC is not efficient with this algorithm, because of the nature of the data.

5 Conclusions

In this work, we have proposed and implemented algorithms for fingerprint identification on two different kinds of parallel platforms: a multi-core server and a Xeon Phi coprocessor.

Our algorithms are based on heaps and exhaustive search, but we also explored an indexing approach using the List of Cluster index. This index did not reach a good performance, mainly because the nature of the vectors did not allow discarding of elements, which is the purpose of the index. This is because the intrinsic dimension of the space is high and it is affected by the well known curse of dimensionality.

Our algorithms achieve up to 97.51x of speed-up, outperforming previous state-of-the-art approaches in GPU and FPGA. Our results show that the computation need of the MCC algorithm can be covered and accelerated using a Xeon Phi. This coprocessor has the advantage of a reduced cost compared to a conventional multi-core server, and also a lower energy consumption. To the best of our knowledge, this is the first work for fingerprint identification using a Xeon Phi coprocessor.

Acknowledgement. This research was partially funded by Project CONICYT FONDEF/ Cuarto Concurso IDeA en dos Etapas del Fondo de Fomento al Desarrollo Científico y Tecnológico, Programa IDeA, FONDEF/CONICYT 2017 ID17i10254. D. Peralta is a Postdoctoral Fellow of the Research Foundation of Flanders.

References

1. Barrientos, R., Gómez, J., Tenllado, C., Prieto, M., Marin, M.: kNN query processing in metric spaces using GPUs. In: 17th International European Conference on Parallel and Distributed Computing (Euro-Par 2011), pp. 380–392 (2011)
2. Barrientos, R.J., Gómez, J.I., Tenllado, C., Matias, M.P., Marin, M.: Range query processing on single and multi GPU environments. Computers & Electrical Engineering **39**(8), 2656 – 2668 (2013)
3. Bellman, R.: Adaptive control processes: a guided tour. A Rand Corporation Research Study Series. Princeton University Press (1961)
4. Bhanu, B., Tan, X.: A triplet based approach for indexing of fingerprint database for identification. In: International Conference on Audio-and Video-Based Biometric Person Authentication, pp. 205–210. Springer (2001)
5. Cao, K., Liu, E., Jain, A.K.: Segmentation and enhancement of latent fingerprints: A coarse to fine ridgestructure dictionary. IEEE transactions on pattern analysis and machine intelligence **36**(9), 1847–1859 (2014)
6. Cappelli, R., Ferrara, M., Maltoni, D.: Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(12), 2128–2141 (2010)
7. Cappelli, R., Ferrara, M., Maltoni, D.: Fingerprint indexing based on minutia cylinder-code. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(5), 1051–1057 (2011)
8. Cappelli, R., Maio, D.: The State of the Art in Fingerprint Classification, pp. 183–205. Springer New York, New York, NY (2004)
9. Chávez, E., Navarro, G.: A compact space decomposition for effective metric indexing. Pattern Recognition Letters **26**(9), 1363–1376 (2005)
10. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. In: ACM Computing Surveys, pp. 33(3):273–321 (2001)
11. Galar, M., Derrac, J., Peralta, D., Triguero, I., Paternain, D., Lopez-Molina, C., García, S., Benítez, J.M., Pagola, M., Barrenechea, E., et al.: A survey of fingerprint classification part i: Taxonomies on feature extraction methods and learning models. Knowledge-based systems **81**, 76–97 (2015)
12. Gil-Costa, V., Barrientos, R.J., Marin, M., Bonacic, C.: Scheduling metric-space queries processing on multi-core processors. In: 18th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP 2010), pp. 187–194. IEEE Computer Society, Pisa, Italy (2010)
13. Gil-Costa, V., Marin, M.: Load balancing query processing in metric-space similarity search. In: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012), pp. 368–375. IEEE, Ottawa, Canada (2012)
14. Gutiérrez, P.D., Lastra, M., Herrera, F., Benítez, J.M.: A high performance fingerprint matching system for large databases based on gpu. IEEE Transactions on Information Forensics and Security **9**(1), 62–71 (2014)

15. Gutierrez, P.D., Lastra, M., Herrera, F., Benitez, J.M.: A high performance fingerprint matching system for large databases based on gpu. IEEE Transactions on Information Forensics and Security **9**(1), 62–71 (2014)
16. Hong, J.H., Min, J.K., Cho, U.K., Cho, S.B.: Fingerprint classification using one-vs-all support vector machines dynamically ordered with naı¨ ve bayes classifiers. Pattern Recognition **41**(2), 662–671 (2008)
17. Jain, A., Flynn, P., Ross, A.A.: Handbook of biometrics. Springer Science & Business Media (2007)
18. Jiang, R.M., Crookes, D.: FPGA-based minutia matching for biometric fingerprint image database retrieval. Journal of Real-Time Image Processing **3**(3), 177–182 (2008)
19. Knuth, D.E.: The Art of Computer Programming, vol. 3. Addison-Wesley (1973)
20. Kumar, A., Kwong, C.: Towards contactless, low-cost and accurate 3D fingerprint identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3438–3443 (2013)
21. Le, H.H., Nguyen, N.H., Nguyen, T.T.: Exploiting GPU for large scale fingerprint identification. In: Asian Conference on Intelligent Information and Database Systems, pp. 688–697. Springer (2016)
22. Lindoso, A., Entrena, L., Izquierdo, J.: FPGA-based acceleration of fingerprint minutiae matching. In: 2007 3rd Southern Conference on Programmable Logic, pp. 81–86 (2007)
23. Maltoni, D., Maio, D., Jain, A., Prabhakar, S.: Handbook of fingerprint recognition. Springer Science & Business Media (2009)
24. Marin, M., Gil-Costa, V.: Approximate distributed metric-space search. In: ACM Workshop on Large-Scale and Distributed Information Retrieval (LSDS-IR 2011). Glasgow, UK (2011)
25. Marin, M., Gil-Costa, V., Bonacic, C., Baeza-Yates, R., Scherson, I.D.: Sync/async parallel search for the efficient design and construction of web search engines. Parallel Computing **36**(4), 153 – 168 (2010)
26. Navarro, G., Uribe-Paredes, R.: Fully dynamic metric access methods based on hyperplane partitioning. Information Systems **36**(4), 734 – 747 (2011)
27. Peralta, D., Triguero, I., Sanchez-Reillo, R., Herrera, F., Benítez, J.M.: Fast fingerprint identification for large databases. Pattern Recognition **47**(2), 588–602 (2014)
28. Partnership for Advanced Computing in Europe (PRACE): Best Practice Guide - Intel Xeon Phi
29. Wang, E., Zhang, Q., Shen, B., Zhang, G., Lu, X., Wu, Q., Wang, Y.: High-performance computing on the intel xeon phi. Springer **5**, 2 (2014)
30. Watson, C.I.: NIST Special Database 14, Fingerprint Database. US National Institute of Standards and Technology (1993)
31. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, *Advances in Database Systems*, vol. 32. Springer (2006)