Exhaustive similarity search on a many-core architecture for finger-vein massive identification

S Guidet\(^1\), R J Barrientos\(^2,3\), R Hernández-García\(^2,4\), and F E Frati\(^1\)

\(^1\) Departamento de Ciencias Básicas y Tecnológicas, Universidad Nacional de Chilécito, Chilécito, Argentina
\(^2\) Laboratorio de Investigación Tecnológica en Reconocimiento de Patrones (LITRP), Universidad Católica del Maule, Talca, Chile
\(^3\) Departamento de Informática e Industrias, Facultad de Ciencias de la Ingeniería, Universidad Católica del Maule, Talca, Chile
\(^4\) Centro de Investigación de Estudios Avanzados del Maule, Universidad Católica del Maule, Talca, Chile

E-mail: sguidet@undec.edu.ar, rhernandez@ucm.cl

Abstract. In massive biometric identification systems, response times mainly depends on the database searching algorithms. Thus, in large databases, an increment in the simultaneous queries traffic becomes a critical factor. This paper proposes an algorithm based on the use of a graphic processing unit to solve the exhaustive similarity search for the mass identification of finger veins, using the binary pattern descriptor of the local vertical line and the Hamming distance. The proposed approach reduces the computation time of the searching process over high query traffic by solving each query with a different processing block. The proposed method allows the identification of individuals in a database of 1 million elements, which is the largest database used for finger-vein identification. Experimental results show that our proposed method resolves up to 28 queries simultaneously (over a database of one million individuals) within a time lower than 3 seconds and achieving a speed-up of 283x. To our knowledge, our work is the first implementation of finger-vein recognition on a general-purpose graphics processing unit, which is the main contribution of this document.

1. Introduction

Massive individuals identification through their unique physiological (fingerprint, face, iris, etc.) or behavioral (voice, gait, signature, etc.) traits using biometric techniques is a problem in modern society. Particularly, finger-vein recognition of individuals has several desirable properties, such as universality, distinctiveness, permanence, and acceptability. The following advantages of this recognition method are reported throughout the literature \cite{1,2}: high accuracy rates, high resistance to criminal manipulation (very difficult to copy or forge), authentication speed, compact size, can only be captured on a live body and does not suffer damage or change over time. Generally, a finger-vein recognition system consists of four main processes: image acquisition, pre-processing, feature extraction, and search. The main contributions in the literature have focused on the process of searching in the biometric database, as this is the stage that requires the most computing consumption.

All biometric methods use unstructured data types. In this type of data, it is not possible to perform an exact search because they do not have a certain structure. Moreover, it is very
difficult to reproduce them under the same conditions (for example: two images taken from the same finger are not equals). Because of this, the result of a search in this kind of database, called identical solution set, will be composed of elements that are similar to the given query, but never equal. Thus, the searching process on a biometric database consists of going through it completely by calculating the similarity between the stored elements and the query.

The similarity search has been widely studied in recent years [3]. This type of search is modeled using metric spaces in large databases. In this model, a distance function that operates on each database element and the query is used to determine the degree of similarity between them [3]. As the size of the database increases, the system must perform more similarity calculations, which makes the calculation time more relevant [4]. Two aspects must be considered about the similarity search process: how the selection of the solution set will be made, and way the database is browsed to apply the first criterion. The k-nearest neighbours (k-NN) algorithm is one of the most widely used methods in the literature in different areas [5]. Similarly, the exhaustive search or brute force algorithm is the most accepted method to go over the database [6, 7]. In some cases it is possible to reduce the computational cost and speed up the search process by using indexes [6, 8, 9]. However, these methods introduce a complex and irregular memory access pattern, making them inefficient in large databases [7].

This article proposes a similarity search method based on a graphics processing unit (GPU) platform for finger-vein individuals identification in massive databases. To solve the similarity search the proposed method implements the k-NN algorithm to select the solution set while scanning the database by brute-force. Our approach provides an adequate response time by using the vertical local line binary pattern (LLBPv) descriptor and the Hamming distance. To our best knowledge, no real application has been proposed for massive finger-vein identification on GPU platforms, which is the main novelty of our approach. In the rest of the paper, the section 2 reviews the concepts and techniques related to finger-vein recognition based on LBP methods, the k-NN searching process, and the GPU architecture. Section 3 introduces the theoretical aspects that support the proposed method and presents the proposed GPU-based scheme for massive individuals identification. Finally, the experimental results and conclusions are given in sections 4 and 5, respectively.

2. Exhaustive search of finger-vein images

Figure 1 shows a general diagram of on finger vein recognition system comprised by four processes. Firstly, a near-infrared (NIR) imaging device (700-1000 nm) captures the veins in the finger. Then, the pre-processing stage improves the image quality and segmentation to obtain the region of interest (called ROI). This step uses a method of extracting ROI that is robust to finger movement and rotation [10]. Also, the limited adaptive histogram equalization technique (CLAHE) is applied to resolve differences in illumination and contrast of the ROI image [11]. The features representing the vein patterns are then extracted from the ROI to be added to the database, using the LLBPv approach [12]. This descriptor improves time, decreases error at the time of identification, and its straight-line shape extracts robust features from images with unclear veins [12]. The following sections describe the searching process in more detail.

2.1. Similarity searching

A similarity search is modeled through a metric spaces \((X, d)\), which consists of a set of \(X\) objects and a distance function \(d\). The distance function determines the similarity between two given objects, which must meet the following properties: strict positivity \((d(x, y) > 0\) and if \(d(x, y) = 0\), then \(x = y\)), symmetry \((d(x, y) = d(y, x))\), and the triangle inequality \((d(x, z) \leq d(x, y) + d(y, z))\) [7].

The bibliography presents different distance functions that can be used, such as cosine distance [3], Hamming distance [12–14], and Euclidean distance [15]. In this work, we use
Hamming distance because it is one of the most used distance functions in the literature. Equation (1) shows the Hamming distance (HD) [12] formula used to obtain the similarity score between the extracted binary code and the stored codes in the database.

$$HD = \frac{||LLBPCodeA \oplus LLBPCodeB||}{LLBPCodeLength},$$

where $\oplus$ is the boolean XOR operator between the corresponding bit pairs of $LLBPCodeA$ and $LLBPCodeB$, which are the binary codes stored in the database and query, respectively. $LLBPCodeLength$ is the total number of bits of the LLBP code of the vein images. The similarity score given by HD is between 0 and 1. Consequently, when two codes correspond to the same finger, the value is close to 0. Conversely, when the codes come from two different fingers, the value tends to 1.

![Flowchart of the proposed approach for massive finger-vein identification based on exhaustive search on GPU.](image)

**Figure 1.** Flowchart of the proposed approach for massive finger-vein identification based on exhaustive search on GPU.

### 2.2. $k$-nearest neighbours querying process

To go through the complete database and obtain the solution set, the $k$-NN method is used because it is one of the most used methods of querying in metric spaces and it is very simple, highly efficient and effective an instance-based classifier [7]. The $k$-NN querying method forms the solution set with the $k$ objects in the database that have the minimum distance between each object and the query element $q$.

When the $k$-NN method is applied, the distance between the query and each element belonging to the database must be calculated. As the size of the database increases, the computation time of the system to provide the solution set also increases. Hence, in massive databases and high-dimensional spaces, exhaustive exploration based on brute-force approaches can become prohibitive. Therefore, we present an GPU-based approach to reduce the computation time for finger-vein massive identification.

### 2.3. Graphic processing unit architecture

The GPU is a parallel coprocessor that can accelerate many compute intensive tasks that follow a data parallel scheme. This section describes GPU aspects that are relevant for achieving an efficient GPU-based algorithm for the $k$-NN problem. GPU parallelism is achieved by using abstract constructs of the programming model, such as CUDA kernel, threads, warps, and CUDA blocks. A CUDA kernel is a program to be executed by the GPU, and it is called from
the Host (as it’s called the CPU). Once it is executed, the threads will carry out the instructions simultaneously, having access to the GPU global memory (VRAM) and caches. The threads are grouped by warps that run together, which is the minimum unit of execution on a GPU (actual warp size is fixed at 32). On the other hand, defining the CUDA block is a requirement to run a CUDA kernel in parallel.

All threads from the same CUDA block can cooperate and share their computed values by doing read/write on shared memory (programmable L1 cache), as well as synchronize among themselves through barriers. Hence, there are two different levels of parallelism: independent threads assigned to different CUDA blocks and cooperating threads within the same CUDA block. An important factor when implementing a CUDA kernel is the multiprocessor occupancy, which is the ratio of active warps to the maximum number of warps supported on a GPU’s multiprocessor [16]. Each GPU’s multiprocessor has a set of \( N \) shared registers available that are allocated among the blocks of threads running on a multiprocessor. Therefore, the CUDA compiler attempts to minimize the use of registers to maximize the number of simultaneous active blocks.

3. Proposed method

The individuals identification process consists of a comprehensive 1:N search, which means comparing the query sample against each record in the database. Each comparison calculates the similarity score between the extracted binary code (i.e. the \( LLBPv \) descriptor) of the query sample and the stored codes by using the Hamming distance. The search procedure returns a list of 32 records with the similarity score in ascending order. We only get the first 32 results because it is the lowest perfect recognition range for \( LLBPv \) with the best precision performance [17].

This process must be done for every query received by the system. Thus, the workload of the system increases with a high rate of queries per unit of time, and the volume of data to be processed increases significantly, so the computation time must be reduced.

3.1. Search algorithm based on graphic processing unit

Aiming to speed up the searching process, our solution solves each query with a different CUDA block, solving multiple queries simultaneously. Through coalescing access, consecutive threads access to consecutive memory addresses, facilitating I/O operations. The entire database is copied to the GPU global memory (DRAM), and for decreasing the read operations latency, the processed scores are moved to the shared memory (Flash L1 type) of each CUDA block.

Each CUDA block processes one query with 128 threads. Each CUDA block reduces the partial results per query by warp. The first thread of each block get the final results (solution set) per query.

Figure 2. Diagram of the proposed GPU-based search algorithm.

Figure 2 shows a diagram of the proposed GPU-based search algorithm. Each GPU thread compares the query allocated to its CUDA block against \( n \) records in the database (where \( n \) is greater than 32) and it builds a heap data structure with the 32 most similar records. Due to the size of each element of the database and the limited size of the shared memory (49 KB), it is only possible to launch 128 threads per CUDA block to guarantee the availability of shared
memory. As a result of this process, each CUDA block provides a partial solution of 32 records each, using one heap per thread as auxiliary structure to store the most similar elements found. Later, taking advantage of the warp size, a further reduction per warp (32 threads of the CUDA block) to 32 heaps of 32 records each is performed in parallel. Finally, the first thread of each CUDA block reduces the 32-heap structure to a single heap with the 32 smallest records in the CUDA block, which is the final solution set for the query solved by the CUDA block.

4. Experimental results

In this section, the performance of the proposed method is evaluated. All experiments were executed on a dedicated server comprised of an Intel(R) Xeon(R) Platinum 8276 CPU @ 2.20GHz, in total 56 physical cores, 39424 KB of L3 Cache, 256 GB of RAM memory, and a GPU NVIDIA Quadro P4000 with 1792 CUDA block and 8GB GDDR5 memory. To evaluate the performance of the proposed algorithm responding to high query traffic, we use the BigFVDB dataset, which was generated in our previous work [17]. For these experiments we use 1.000.000 samples, due to operating with a larger number of samples exceeds the intended scope. It should be noted that a query in BigFVDB can only have one possible result, but it performs 1:N matching comparisons.

The experiments were performed by increasing the number of available queries on the system, starting with 1 up to 200 by increasing in power of two. To obtain an unbiased result and to guarantee the stability of the results, the time measurements were averaged by repeating each test 100 times. Besides, in all experiments, it was checked that the same results were obtained for the same comparisons.

To evaluate the performance of the proposed solution, we compared the GPU version and the multi-core version called Round-Robin distribution (dist-RR), that obtained the best performance in a previous research [17]. Figure 3 shows the times in seconds for the multi-core version dist-RR and the proposed GPU-based algorithm, called K-NN-GPU, for an increasing number of queries. It can be seen the superiority (averaged speed-up of 10x) of the implementation on GPU compared to the multi-core implementation, which was executed with 32 processing CPU-threads, and each thread executed over a different core.

![Figure 3. Comparison of the processing time obtained by the multi-core algorithm [17] vs the proposed GPU-based approach by varying the number of simultaneous queries.](image)

Figure 4 zoom up the results obtained for the GPU version only. It can be seen that as the number of simultaneous queries to be solved increases, the system shows a staggered behavior. This behavior is because the used GPU model can only have 28 active CUDA blocks simultaneously. The number of active CUDA blocks is calculated according to the GPU occupancy data. Hence, taking into account that, there are 14 GPU multiprocessors with 256
active threads per multiprocessor, and the GPU kernel runs with 128 threads per CUDA block, then the GPU has 12.5% occupancy factor.

Moreover, it should be noted that the system is capable of solving 28 queries simultaneously in a response time of less than 3 seconds, achieving a maximum speed-up of 283x. The speed achieved by the proposed algorithm was calculated as the ratio of the sequence algorithm run time on CPUs to the GPU algorithm run time executed with multiple CUDA blocks.

![Figure 4. Results of processing time obtained by the proposed GPU-based approach by varying the number of simultaneous queries.](image)

5. Conclusions
This paper presents a GPU-based implementation for massive finger-vein individuals identification. The proposed method aims to reduce the computation time of the searching process over high query traffic. Our proposal solves each query with a different CUDA block, which builds a heap data structure with the 32 most similar records. Thus, the GPU-based algorithm computes as many queries as active CUDA blocks the GPU can maintain.

Experimental validation shows that the multi-core approach obtains a linear behaviour as the workload increases. On the other hand, the GPU implementation has a staggered behavior as the number of simultaneous queries to be solved increases. When comparing both, results show that the GPU version is approximately 10 times better than the multi-core version. The results show that the system can process up to 28 queries simultaneously with a response time of less than 3 seconds, achieving a maximum speed-up of 283x.

Future work plans to evaluate the proposed approach by increasing the number of individuals in the database, to reach 16 million of records. Using a database of that size faces the issue of the overall memory capacity of the GPU and exploring a multi-GPU approach.

References
[1] Miura N, Nagasaka A and Miyatake T 2004 Feature extraction of finger-vein pattern based on repeated line tracking and its application to personal identification Machine Vision and Applications 15(4) 194–203
[2] Ezhilmaran D, Joseph P R B 2015 A study of feature extraction techniques and image enhancement algorithms for finger vein recognition International Journal of PharmTech Research 8(8) 222–229
[3] Hernández-García R, Barrionte R J, Rojas C and Mora M 2019 Individuals identification based on palm vein matching under a parallel environment Applied Sciences 9(14) 2805
[4] Dos Santos, E, Osiris Sofia A A, Uribe Paredes R 2015 Procesamiento de búsquedas por similitud. Tecnologías de Paralelización e Indexación Informes Científicos Técnicos-UNPA 7(2) 107–136
[5] Barrientos R J, Gómez J I, Tenllado C, Matias M P and Marin M 2013 Range query processing on single and multi gpu environments Computers & Electrical Engineering 39 2656–2668
[6] Arefin A S, Riveros C, Berretta R and Moscato P 2012 Gpu-fs-k nn: A software tool for fast and scalable k nn computation using gpus PLOS ONE 7 e44000
[7] Barrientos R J, Millaguir F, Sánchez J L and Arias E 2017 Gpu-based exhaustive algorithms processing k nn queries The Journal of Supercomputing 73 4611–4634
[8] Böhm C, Berchtold S and Keim D A 2001 Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases ACM Computing Surveys (CSUR) 33(3) 322–373
[9] De Battista A, Herrera N E, Retamal G A G and Pascal A 2010 Metric-temporal access methods Journal of Computer Science and Technology 10(02) 54–60
[10] Yang L, Yang G, Yin Y and Xiao R 2013 Sliding window-based region of interest extraction for finger vein images Sensors 13 3799–3815
[11] Reza A M 2004 Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology 38 35–44
[12] Rosdi B A, WShing C and Suandi S A 2011 Finger vein recognition using local line binary pattern Sensors 11 11357–11371
[13] Lee E C, Lee H C and Park K R 2009 Finger vein recognition using minutia-based alignment and local binary pattern-based feature extraction International Journal of Imaging Systems and Technology 19 179–186
[14] Lee E C, Jung H and Kim D 2011 New finger biometric method using near infrared imaging Sensors 11 2319–2333
[15] Kang W, Liu Y, Wu Q and Yue X 2014 Contact-free palm-vein recognition based on local invariant features PLOS ONE 9 e97548
[16] Toledo C M, Barrientos R J and Ávila A I 2016 Similarity (range and knn) queries processing on an intel xeon phi coprocessor Cluster Computing 19 57–71
[17] Hernández-García R, Guidet S, Barrientos R J and Frati F E 2019 Massive finger-vein identification based on local line binary pattern under parallel and distributed systems 38th International Conference of the Chilean Computer Science Society (SCCC) (Concepción: IEEE)