Finger-vein individuals identification on massive databases

Sebastián Guidet[1], Ricardo J. Barrientos[2,3], Fernando Emmanuel Frati[1], and Ruber Hernández-García[2]

[1] Department of Basic and Technological Sciences,
Universidad Nacional de Chilecito, Chilecito, La Rioja, Argentina
sguidet@undec.edu.ar, fefrati@undec.edu.ar
[2] Laboratory of Technological Research in Pattern Recognition (LITRP),
Faculty of Engineering Sciences, Universidad Católica del Maule, Talca, Chile
rbarrientos@ucm.cl, rhernandez@ucm.cl
[3] Department of Computer Science and Industries, Faculty of Engineering Science,
Universidad Católica del Maule, Talca, Chile

Abstract. In massive biometric identification, response times highly depend on the searching algorithms. Traditional systems operate with databases of up to 10,000 records. In large databases, with an increasing number of simultaneous queries, the system response time is a critical factor. This work proposes a GPU-based implementation for the matching process of finger-vein massive identification. Experimental results show that our approach solves up to 256 simultaneous queries on large databases achieving up to 136x.

Keywords: High Performance Computing, identification of individuals, local linear binary pattern, finger veins, GPU.

1 Introduction

Massive identification of individuals by using biometric techniques is a difficult problem in modern society. Particularly, vein-based biometric provides universality, distinctiveness, permanence, and acceptability. In the literature, different approaches based on finger-vein recognition [4] report several advantages of the finger-vein biometrics, such as high accuracy, high resistance to criminal manipulation (very difficult to copy or forge), authentication speed, compact size, liveness detection, and does not suffer damage or change over time.

On the other hand, the searching process on a biometric database consists of an exhaustive searching by calculating similarity metrics between the stored elements and the sample to be identified. As the size of the database increases, the identification accuracy decreases, while the recognition process execution time increases significantly. Besides, a high rate of queries per unit time is to be expected in this type of system. Thus, it is essential to ensure a reasonable response time [1].

This paper proposes a searching method based on GPGPU (general-purpose computing on graphics processing units) for finger-vein individuals identification on massive databases. Our approach guarantees an adequate response time by using the Vertical Linear Binary Pattern (LLBPv) descriptor and the Hamming distance as a similarity function. As far as we know, no real application has been proposed for finger-vein massive identification on GPU platforms, which is the main novelty of our approach.

2 Finger-vein identification system

A finger-vein identification system comprises of four main processes. Initially, a near-infrared (NIR) imaging device (700-1000 nm) captures an image of the finger-vein patterns. Later, the pre-processing stage obtains the region of interest (ROI) and enhances image quality. For ROI segmentation, we adopt the method proposed in [5], which is robust to finger movement and rotation. Also, the limited adaptive histogram equalization technique (CLAHE) is applied to adjust differences in illumination and contrast [6]. During the feature extraction process, the final enhanced image is represented by using the Vertical Local Line Binary Pattern ($LLBPv$) descriptor. This descriptor decreases the computation-time and its straight-line shape extracts robust features from images with unclear veins [3]. Finally, the searching process is performed on the database by using a similarity function. The following section describes this last procedure in more detail.

3 Searching process on a massive database

Individuals identification consists of a 1:N exhaustive searching on the database, which means comparing the individual's sample against each record of the database. Each comparison computes a similarity score between the extracted binary code (i.e LLBPv descriptor) and the stored codes. The Hamming's distance similarity function is adopted for this calculation due to its effectiveness for comparing binary codes [3]. When two codes correspond to the same finger, the similarity score tends to be 0. Instead, if the codes are from different fingers, the value is closer to 1.

The searching procedure returns a list of 32 records sorted by the similarity score in ascending order. We only obtain the first 32 results because it is the lowest perfect recognition range for $LLBPv$ with the best accuracy performance [2].

This process must be performed for every query received by the system. Thus, the workload of the system increases with a high rate of queries per unit time, and the data volume to be processed increases significantly, therefore the response time should be reduced and to use a GPU is a suitable solution.

3.1 GPU-based searching algorithm

Aiming to speed up the computation time of the searching process, our proposed solution divides the similarity calculation tasks among all GPU threads. Through coalescing access, consecutive threads access to consecutive memory addresses, facilitating I/O operations. The entire database is copied to the global memory (DRAM), and for decreasing the read operations latency, the records to be processed are moved to the shared memory (Flash L1 type) of each CUDA Block.

Each GPU thread makes comparison calculations between the query to be identified and the records on the database. Our proposed algorithm uses the shared memory to store the query and also a set of *heap* data structure, and this implies that it is only possible to execute 128 threads per CUDA Block in our kernel. Each GPU thread keep the lowest distances found in its heap stored in shared memory. We choose a heap data structure because its efficiency in the insertion and extraction operations. As a result of this process, each CUDA Block reduces the database to 128 heaps of 32 records each. Then, taking account that a warp (32 threads) is the minimum execution unit in GPU, the threads of the first warp of each CUDA Block access the data previously found, and reduce them to elements stored in 32 heaps in shared memory. Finally, the first thread of each CUDA Block reduce the elements of the 32 previous heaps to just one heap with the 32 lower elements, also stored in shared memory. Each CUDA Block transfers its 32 elements to CPU, where all of them are merged, and a sequential quicksort algorithm is performed over these elements to select the lowest final distance.

When the system replies to multiple simultaneous queries, it repeats the whole process described for each query request.

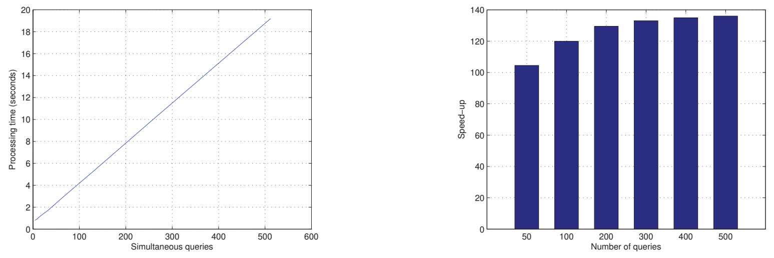
4 Experimental work

The experimental environment consists of a GPU NVIDIA GeForce GTX 1080 Ti with 3584 CUDA cores and 11GB GDDR5X memory, and the host computer is composed of 2 × Intel Xeon Gold 6140 CPU @ 2.30GHz, in total 36 physical cores, 24.75MB L3 cache and 128GB RAM.

To evaluate the performance of the proposed algorithm responding to a high query traffic, we use the BigFVDB dataset, which was generated in our previous work [2]. For these experiments we used 339,968 samples, due the capacity limits on the GPU global memory. 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 4 up to 512 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, it was checked that in all experiments the same results were obtained for the same comparisons. In all cases, the maximum expected response time was 10 seconds (time threshold), which was defined based on previous evaluations [2]. However, it could vary according to the system requirements and the size of the database to be processed. Figure 1 summarizes the obtained results. It is worth highlighting that the proposed method solves up to 512 queries while keeping the response time less than the time threshold, keeping a very similar time in solving each different query.

It should be clarified that for the calculation of speed - up (Time(Sequential)/ Time(Parallel)) in the Figure 1. (b) the time of execution of the sequential version in CPU was taken as reference.



(a) time of the proposed GPU-based (b) Speed-up

Fig. 1: Running time of our proposed GPU-based algorithm and its speed-up solving different quantity of queries.

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 a high query traffic. We used a set of heaps as auxiliary structures to keep the lower elements found, and we also propose an algorithm in GPU to reduce the elements of the heaps to obtain the final query result.

The experimental validation shows that the proposed approach obtains a linear behavior as the workload increases. The proposed method keeps response times lower than 10 seconds with an increasing number of simultaneous queries, without requiring to increase the involved hardware resources, achieving up to 136x of speed-up solving 500 queries.

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

References

1. Cappelli, R., Ferrara, M., Maltoni, D.: Large-scale fingerprint identification on gpu. Information Sciences 306, 1–20 (2015)
2. Hernández-García, R., Guidet, S., Barrientos, R.J., Frati, F.E.: Massive finger-vein identification based on local line binary pattern under parallel and distributed systems. In: 2019 38th International Conference of the Chilean Computer Science Society (SCCC). pp. 1–7. IEEE (2019)
3. Rosdi, B.A., W.Shing, C., Suandi, S.A.: Finger vein recognition using local line binary pattern. Sensors 11, 11357–11371 (2011)
4. Shaheed, K., Liu, H., Yang, G., Qureshi, I., Gou, J., Yin, Y.: A systematic review of finger vein recognition techniques. Information 9(9), 213 (2018)
5. Yang, L., Yang, G., Yin, Y., Xiao, R.: Sliding window-based region of interest extraction for finger vein images. Sensors 13(3), 3799–3815 (2013)
6. Zuiderveld, K.: Contrast Limited Adaptive Histogram Equalization. In: Heckbert, P.S. (ed.) Chapter VIII.5, Graphics Gems IV, pp. 474–485. Academic Press Professional, Inc. (1994)