Parallel Implementation of Fingerprint Identification for Large Databases

Anastasija Vasilevska

November 2020

1 Related Work

Fingerprints are the most widely deployed biometric characteristics as stated in the "Handbook of Fingerprint Recognition" [1]. Every forensics and law enforcement agency uses fingerprint identification systems [2]. Unfortunately, the way the matching works makes it very hard to get fast and accurate results when it comes to large databases.

That's why, the goal is to achieve higher speed matching without losing accuracy. That could ideally be achieved by optimizing execution time of fingerprint minutiae-based feature extraction using parallel process [3]. The results of this performance are consistent, i.e. faster about 57, 55 and 60 percent for each tested database.

One possible solution is using a GPU fingerprint matching system [4] based on the MCC algorithm [5]. Raffaele Cappelli [6] introduces a new parallel algorithm also using GPUs to solve the problem. Ali Ismail Awad [7] worked on using detectors such as SIFT and SURF, which are running on CPU and GPU, proving the GPU to be the one with better results.

Another solution is proposed in the "Efficient Fingerprint Matching Using GPU" [8] article where the authors suggest mapping a generalised minutia neighbour-based novel encoding and matching algorithm on low-cost GPU technology.

There is also a parallel fingerprint classification system from 1993 [9] that uses image-based ridge-valley features, K-L transformations, and neural networks to perform pattern level classification.

A very important step for the fingerprint identification system is fast and reliable "thinning" of the image, in other words to skeletonize the fingerprint image for minutiae extraction [10].

Parallel implementation of a search algorithm can also be used for an audio fingerprinting system. Compared to the CPU only implementations, the proposed GPU implementation [11] reduces run times by up to 150 times for one intersection algorithm and by up to 379 times for the another intersection algorithm.

2 Solution Architecture

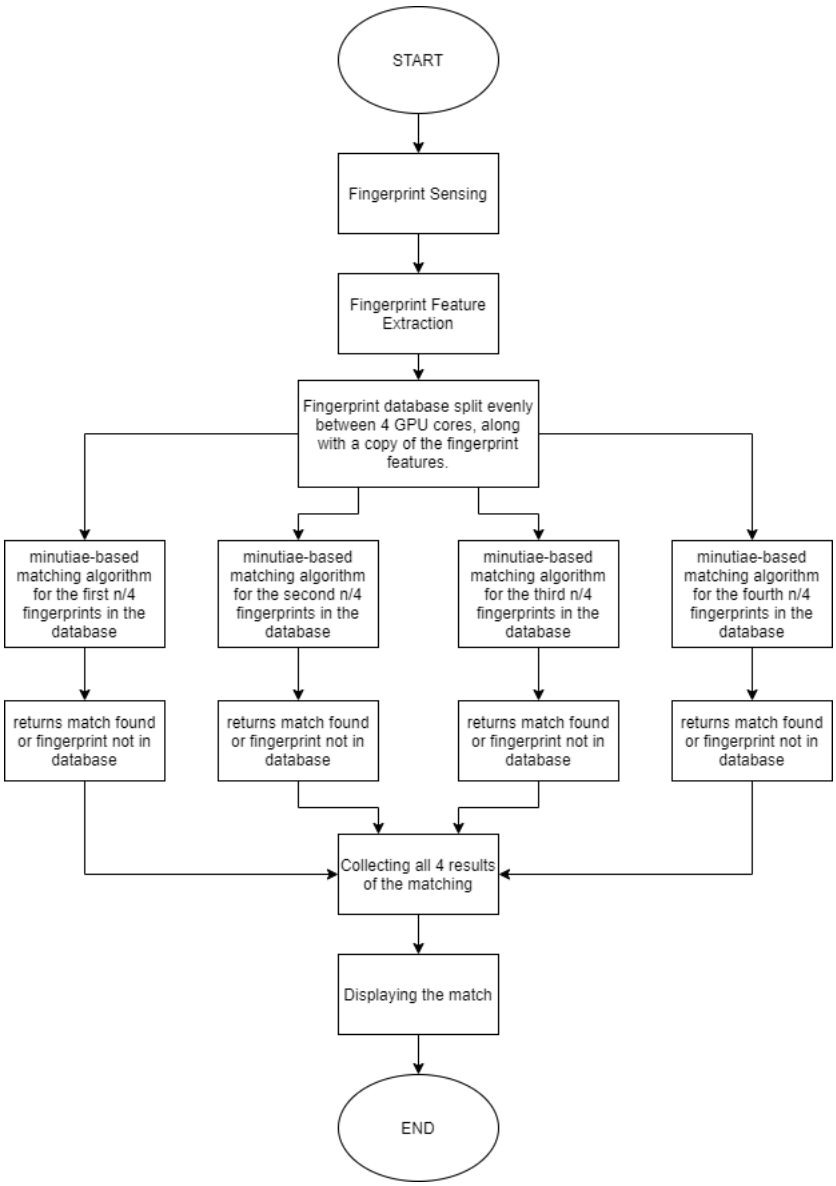

Figure 1: Diagram of the proposed solution for a parallel implementation of fingerprint identification

The fingerprint identification process consists of 3 main steps: fingerprint sensing, feature extraction and matching.

The fingerprint sensing process includes a live-scan of a person's finger with an electronic fingerprint scanner. The scanned digital photo needs to fit a few specifications for quality and format of fingerprint images.

The next step of the fingerprint identification process is the feature extraction step. The feature extractions involves filtering the image and using various algorithms to find and extract the singularities and minutiae of the fingerprint. Singularities are regions of the fingerprint where the ridges make distinctive shapes also known as a loop, delta (arch) or whorl.

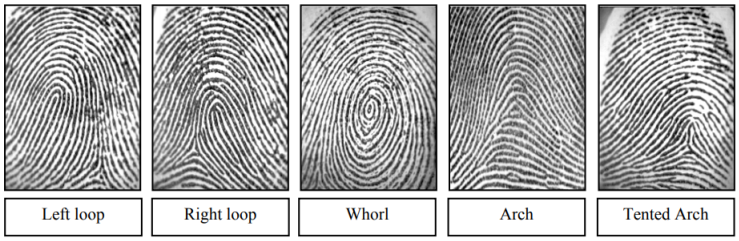

Figure 2: Fingerprint Singularities

Fingerprint minutiae are the different ways the ridges can be discontinuous. The two main types are termination, when the ridge comes to an end, and bifurcation, when the ridge divides into two ridges.

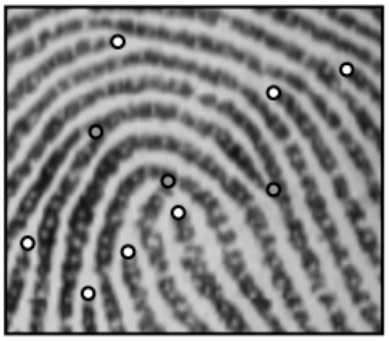

Figure 3: Termination (white) and bifurcation (gray) minutiae in a sample fingerprint

The last step of the process is the fingerprint matching. Matching means that the singularities and minutiae need to be compared to every fingerprint in the database until a match is found. This is the step that the parallel implementation will focus on. The proposed idea is using flat-data parallelism on a GPU with multiple cores. In the diagram the example has 4 cores. The database will be split evenly between all the cores so that each core has about 1/4 of the whole

database to work on. A copy of the fingerprint features is used in all 4 cores. This means that the matching process should be about 4 times faster than the worst case scenario in a serial implementation. The main problem with this system is that it always takes the same amount of time for all the cores to finish the matching process. This means there is a chance that the serial implementation could be faster in certain situations, mainly if the match is found somewhere in the first 1/4 of the database. The chances of this situation happening become smaller as the number of cores rises. In a hypothetical situation with 100 GPU cores that edge case would only be in the first 1/100, 1000 cores only in the first 1/1000 and so on. This means that the parallel implementation should produce faster matching times than a serial implementation in most situations.

After the matching processes are done, all of them return the results they've gotten. With a clean, quality fingerprint image, 3 out of the 4 processes should return that the fingerprint was not found in the database and the 4th one should return the match. All of the results are collected and the match found is displayed.

3 Results

The experiment results show that the parallel method significantly speeds up the computing time of the fingerprint identification process. The experiment was made with the FVC2000 databases that include up to 880 available fingerprint images. Subsets of 50, 100, 500 and 880 images were used and the experiment was done with 1, 2 and 4 instances.

The table with the results from the experiment can be seen on Figure 4. While the difference isn't that big on the smaller subsets of the database, the improvement becomes more visible and significant with the larger subsets, even cutting down on hours of work.

Fingerprint Database Size:	50	100	500	880
Time:	83s	169s	4126s	12451s
Time with 2 instances:	68s	113s	3487s	8265s
Time with 4 instances:	37s	59s	1969s	5043s

Figure 4: Table of results for single and parallel methods

Figure 5 shows the comparison between the single method and the parallel method with 2 instances. When used on the largest subset of data, the graph shows that the parallel method takes only 2/3 of the time the single method does.

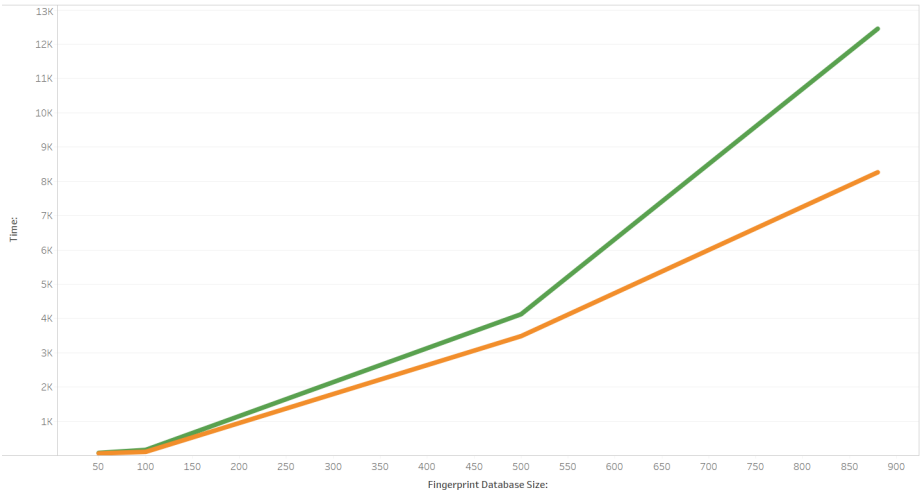

Figure 5: Graph for 1 instance vs 2 instances

Figure 6 shows the comparison between the single method and the parallel method with 4 instances, and here the parallel method takes about 1/2 of the time the single method needs.

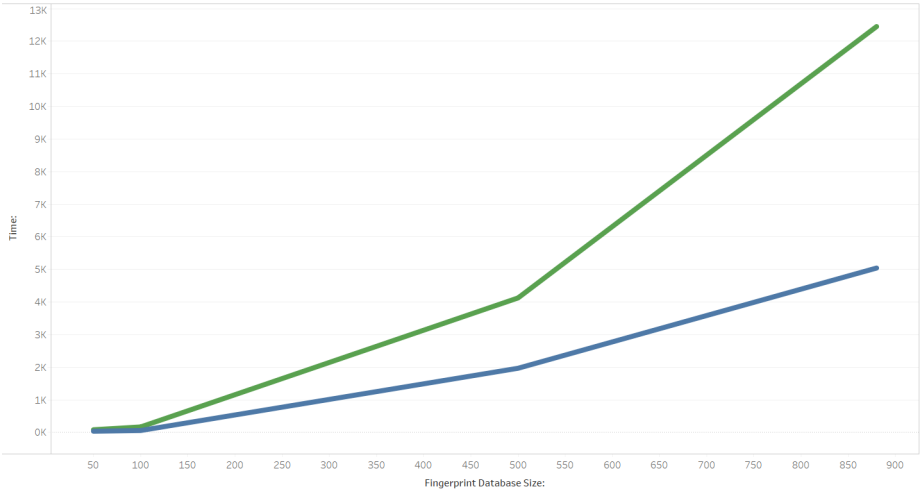

Figure 6: Graph for 1 instance vs 4 instances

Figure 7 shows the comparison between the parallel methods with 2 and 4 instances.

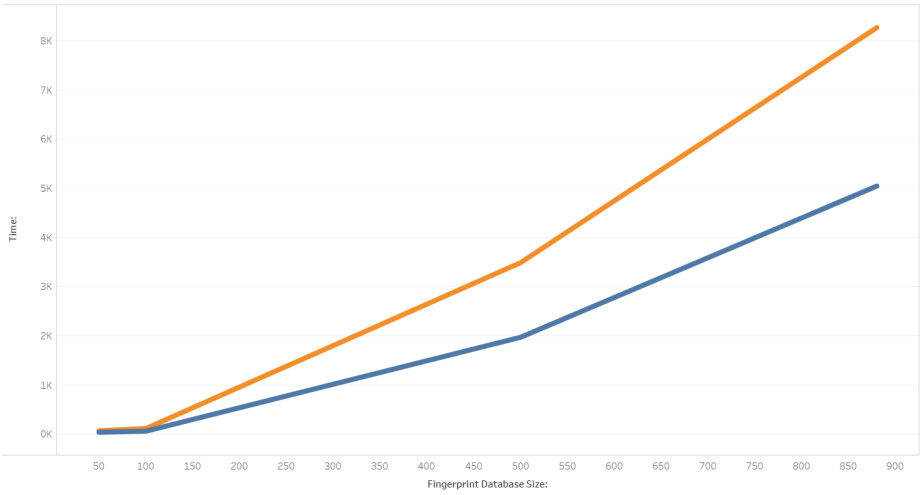

Figure 7: Graph for 2 instances vs 4 instances

References

[1] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. 2003.

[2] D. Maltoni. A tutorial on fingerprint recognition. *Advanced Studies in Biometrics*, 2003.

[3] G. Indrawan, B. Sitohang, and S. Akbar. Parallel processing for fingerprint feature extraction. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–6, 2011.

[4] P. D. Gutiérrez, M. Lastra, F. Herrera, and J. M. Benítez. A high performance fingerprint matching system for large databases based on gpu. *IEEE Transactions on Information Forensics and Security*, 9(1):62–71, 2014.

[5] R. Cappelli, M. Ferrara, and D. Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Transactions on Software Engineering*, 32(12), 2010.

[6] R. Cappelli, M. Ferrara, and D. Maltoni. Large-scale fingerprint identification on gpu. *Information Sciences*, 306:1–20, 2015.

[7] A. I. Awad. Fingerprint local invariant feature extraction on gpu with cuda. *Informatica*, 37:279–284, 2013.

[8] M. Ghafoor, S. Iqbal, S. A. Tariq, I. A. Taj, and N. M. Jafri. Efficient fingerprint matching using gpu. *IET Image Processing*, 12(2):274–284, 2018.

[9] C. L. Wilson, G. T. Candela, and C. 1. Watson. Neural network fingerprint classification. *Artificial Neural Networks*, 1(2), 1993.

[10] H. Xu, Y. Qu, Y. Zhang, and F. Zhao. Fpga based parallel thinning for binary fingerprint image. In *2009 Chinese Conference on Pattern Recognition*, pages 1–4, 2009.

[11] C. Ouali, P. Dumouchel, and V. Gupta. Gpu implementation of an audio fingerprints similarity search algorithm. In *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2015.