Biostatistics and Biometrics
Open Access Journal
ISSN: 2573-2633

Review Article
Volume2 Issue 1 - May 2017
DOI: 10.19080/BBOAJ.2017.02.555577

Biostat Biometrics Open Acc J

GPU Processing for Biometric Big Data Based Identification. Why and what for?

Miguel Lastra[1*], Pablo D Gutiérrez[2], José M Benítez[2] and Francisco Herrera[2]

[1]Department of Software Engineering, University of Granada, Spain

[2]Department of Computer Science & Artificial Intelligence, University of Granada, Spain

Submission: April 3, 2017; **Published:**May 25, 2017

***Corresponding author:** Miguel Lastra, Department of Software Engineering, University of Granada, Spain, Tel: +34 958246144;
Email: mlastral@ugr.es

Abstract

The identification of people by means of biometric features has been an active research topic before computers started being used for such tasks. The scale of problems than can be addressed by the use of personal computers or computer clusters has grown but there are still real-world scenarios where the biometric features database cannot be processed in real time.

This paper tackles the use of graphics processing units (GPUs) which are high performance and cost effective computing elements, as building blocks of large-scale identification systems. A review of the existing GPU based systems is introduced focusing on GPU based fingerprint matching systems. The impressive results obtained by GPU-based systems are only possible by applying a careful biometric matching algorithm redesign, task granularity, asynchronous memory transfers and task overlapping techniques.

Keywords: Biometry; GPU; Identification; Fingerprint; Matching

Introduction

The problem of unequivocally identifying people by biometric features has been an open issue during centuries. Humans possess characteristics associated to different body parts and behaviors which are known to be different on every human being (even on identical twins) like fingerprints, footprints, iris textures, etc. Being able to identify people by using one or a set of these features has been studied in forensics and crime investigation environments motivated by the need to determine the authors of criminal acts and to identify bodies [1].

Nowadays, due to the increasing security requirements, being able to establish the identity of one person is a hot topic in airport passenger identification tasks, access control systems, payment systems, etc. In many of these large-scale environments, it is necessary to identify a person considering a user base of millions or tens of millions. Tackling the identification of people in such domains imposes requirements on the identification methods not only related to accuracy but also on efficiency. The design of effective and efficient identification systems requires using high performance hardware platforms and therefore software specifically designed and suited to those platforms.

In the area of high performance computation, Graphics Processing Units (GPUs) [2] have emerged as a very powerful and cost effective solution and are being widely used in many physics simulation systems. These devices can also be applied in biometric systems to widen the applicability field of existing methods and also to favor the design of new algorithms motivated by the new performance levels that can be obtained at a reasonable cost. This work presents a study on the use of GPUs in biometric identification tasks and some successful applications on fingerprint identification in large-scale environments.

The structure of this paper is as follows: First, the topic of general purpose computation based on GPUs is introduced. Then we review the existing works where GPUs are applied to biometric systems. Finally the conclusions are presented.

Biometric Big Data Processing Based on GPU. Why?

Graphics Processing Units (GPUs) are devices that were designed to efficiently process the graphics pipeline that allows producing computer generated 2D and 3D graphics. Early GPUs had a set of fixed functions and were barely programmable. This situation changed rapidly and the programmability of these devices increased and researchers started using them to perform other tasks than processing vertices and pixels. The increasing computational power required by the video game industry and the development of general purpose frameworks like CUDA [3] or Open CL [4], has provided the research community with coprocessors of unprecedented computing power in the domain of personal computers. The parallel processing architecture of current GPUs reaches the order of 11 Teraflops available in a single card at a price of around 800€ (NVIDIA GTX 1080Ti). As the GPU based scientific computation has been increasingly adopted, companies like NVIDIA [5] have released GPU based cards that no longer offer any display output and are only focused on high performance computation.

The impressive computational power of current GPUs can only be taken advantage of by using parallel software design and programming techniques. As an example, the hardware architecture of the NVIDIA GTX 1080Ti is composed of one GPU that is composed of 28 multiprocessors (SMs). Each of these multiprocessors has 128 computing cores which yields a total of 3584 cores. These cores are not general purpose oriented like those in a CPU and can be considered as ALUs. This architecture works in a SIMD (Single Instruction Multiple Data), or SIMT (Single Instruction Multiple Threads) fashion, which is suitable for problems with a high data parallelism structure where the same instruction can be run on different data elements in a parallel way. The programs to be run on a GPU are called kernels. They represent the code that will be run in parallel by a number of threads. To fulfill a computational task it might be necessary to run a set of kernels. All the threads created by a kernel launch will run the same code but applied to different data elements of the complete dataset.

Taking into account the high number of processing elements and that even the high-speed GPU memory is still several orders of magnitude slower than the computing components, efficiently using the available resources and hiding memory latencies are key aspects to obtain peak performance. The main challenges when designing GPU based programs are mapping the problem to a set of parallel threads and to hide memory latencies. It is important to find the right balance between thread level parallelism and instruction level parallelism. The first type of parallelism is achieved by running many threads in parallel and the second one by providing enough computational workload to each thread to maximize the instructions that each thread can run before a memory access has to be performed.

In the CUDA framework, at the lowest level, threads are grouped into warps that are groups of 32 threads that are run in a warp-synchronous way. This means that it is important to avoid any code divergences inside a warp as this would lead to execution serialization. At a higher level, threads are also grouped into blocks. Each thread block is guaranteed to be run on the same multiprocessor, which allows sharing resources like a high speed shared memory and using explicit synchronization mechanisms (although its use should be minimized to maximize the parallelism). The set of blocks that are created form a grid, which holds the total number of threads associated to a kernel.

Each multiprocessor has a high-speed memory area which can be used as a local cache and also as a region where threads of the same block can interchange data. There is also a large register set which is used by the threads run on that multiprocessor. At the level of thread-level parallelism, memory latencies are hidden by supplying a large amount of threads to each SM. By using the large set of registers, many warps are kept in a ready to run state and when a warp requires a memory access, it is stopped and another warp is run. Context switches do not require any register data to be stored in memory or state to be reloaded to any registers and are therefore very lightweight. At the level of instruction-level parallelism, keeping the instruction pipeline of each core as full as possible is also very important. The computational resources are used more effectively by supplying enough workload to each thread and maximizing the number of instructions that can be run independently of memory accesses. As threads get more complex, their register and shared memory requirements might increase which can potentially reduce the thread level parallelism but too lightweight threads would exhibit a low instruction level parallelism. It is therefore essential to find the best task granularity assigned to threads to achieve a tradeoff between both types of parallelism. Finally, as CPU and GPU memory areas are only physically connected by the PCIe bus, memory transfers to and from the GPU may have a high time cost compared to the computation phases. In order to avoid any bottlenecks related to memory transfers, these can be performed asynchronously and in parallel with respect to kernel launches.

Biometric Big Data. What?

GPUs have been widely used in scientific and simulation software even before general purpose computation frameworks existed, but in the field of biometrics its use is still scarce. Taking into account the origins of GPU based computations it is no surprise that they have been mainly used in systems which are completely or partially image based. GPUs process 2D images very efficiently by design and this is therefore an area of natural applicability.

GPU devices offer the opportunity of tackling large-scale scenarios in Biometrics. Initial identification use cases with several hundreds of users (worker identification, membership verification in gyms, etc.) have been superseded by use cases like concert or sport event attendees identification, where tens of thousands of people need to be identified in real time; or even country-scale scenarios with millions of users. Traditional techniques do not necessarily lead to efficient (in terms of performance and cost) and scalable solutions when the size of the data to process increases by orders of magnitude. GPUs provide the ability to successfully approach this Big Data problems to build space-constrained and energy consumption effective solutions.

This section presents the GPU based state of the art in Biometrics. First fingerprints, which are the current main feature, are considered and then other biometrics such as palmprints, iris textures and faces are tackled. In the area of fingerprint identification, in [6] a GPU based work for enhancing fingerprint images by using a Gabor filter bank and a GPU based FFT implementation was shown. In [7] a GPU and image based saliency detection algorithm is presented by using the GPU to apply convolution kernels. Focusing on the matching process, in [8] two image based fingerprint matching algorithms are optimized by using a GPU based library but the quality of the extracted features and some problems due to memory transfers are not addressed. These works have in common that the hardware that was used is far from the state of the art GPU hardware and that scenarios with large fingerprint databases are not tackled.

The work presented by [9] is the first to show the design of a GPU matching system suitable for very large-scale environments and based on a matching technique of proven quality. The GPU-based system of this work, running on a computer equipped with 4 GPUs, is on pair of a computer cluster of 12 dual-processor nodes (2 Intel Xeon E5-2620 processors per node, each one with 6 physical cores) when the results are compared to the ones presented by [10]. Both works use the well-known fingerprint matching technique called the Minutia Cylinder Code (MCC) [11] which achieved ERR rates of 0.57% and 2.31% in the standard and hard tests respectively of the FVC onGoing competition [12]. The main disadvantage of MCC is that it is computationally demanding. Another work that addressed the redesign of a fingerprint matching technique to make it suitable for GPU devices and large-scale scenarios is the one presented by [13]. In this work a less computationally intensive algorithm is used at the cost of lower accuracy rates. These GPU-based systems achieve rates from tens of thousands to over 1 million fingerprints processed per second with a computer equipped with a number of GPUs up to four units.

In a related fashion, is the paper [14] that presents another highly optimized version of the MCC algorithm for GPUs. In this case the binary version of this algorithm was redesigned. Many floating point operations are substituted by bitwise operations which allow many similarity operations to be packed in a single bitwise operation. Thanks to a careful design and four GPUs, a throughput of up to 35 million fingerprints per second was achieved.

In the field of specialized hardware, FPGAs (field-programmable gate arrays) have also been used for creating fingerprint matching systems. In [15] and [16] low cost proposals are presented that could be used in embedded system for small-scale scenarios more suited for verification systems than identification ones. The authors of [17] proposed another FPGA matching algorithm and their results show the processing of over 1 million fingerprints per second with a speed-up factor of 45 with respect to a reference CPU version. The matching algorithm is really simple and does therefore not account for rotations or distortions (it requires an alignment preprocessing step) and the reference CPU hardware is very entry level (Intel Celeron) to be used as reference in terms of performance. The authors do not provide any information about the accuracy of their system.

The use of palmprints for identification has also been studied and GPU-based systems have been presented [18]. The matching system of this work achieves around 3500 matching operations per second on what currently can be considered as below commodity hardware. The authors used database sizes up to 600000 samples. On the other hand, there are several works in the field of face recognition. A system assisted by GPUs using a GPU based MATLAB toolbox can be found in [19] and GPUs applied on the processing of normal maps obtained from face images in [20]. In both cases, due to the low-spec and outdated hardware that has been used it is difficult to obtain an estimation about the performance in large environments with current hardware. The work in [20] is also limited by the restriction of the use of the graphics pipeline. Recently several GPU implementations of a very well-known technique (Eigenface) based on PCA (Principal Component Analysis) have been presented [21-23].

Another relevant biometric feature is the iris texture. The authors of [24,25] use GPUs to perform the matching of iris pairs based on the Hamming distance of iris codes. In the first work the authors achieve up to over 40 million matching operations per second (4 million when head tilt is taken into account) and, in the second one rates of up to 1 million matching operations per second are reached. In [26] the GPU is applied only to the iris segmentation phase. Additional approaches to iris-based matching have been presented by [27] where the energy consumed is also taken into account and [28] where very modest GPU hardware outperforms a multithreaded CPU based solution.

Concluding Remarks

This paper has presented the main existing contributions to the state of the art in GPU-based biometric feature matching. Early attempts did only focus on achieving a relevant speed-up ratio, but recently several GPU software designs, suited for real large-scale scenarios, have been presented in the field of fingerprint matching. These works prove GPUs are very well-suited for this type of systems when compared to classic computer clusters. GPUs reduce the cost and favor high-density hardware solutions. GPUs have the potential of being the key stone of future identification systems at country-scale. These systems could be multimodal (using several features for each identification process) and be used as fast and reliable identity verification systems for concert attendees, hospital patients, airport passengers, etc.

Acknowledgement

This work was supported by the Spanish National Research Projects TIN2013-47210-P, TIN2014-57251-P and TIN2016-81113-R and by the Andalusian Regional Government Excellence Research Project P12-TIC-2958.

References

1. Maltoni D, Maio D, Jain AK, Prabhakar S (2009) Handbook of fingerprint recognition. Springer-Verlag New York Inc, USA.
2. Jason Sanders, Edward Kandrot (2010) CUDA by Example: An Introduction to General-Purpose GPU Programming (1st edn). Addison-Wesley Professional, Boston, pp. 1-3111.
3. http://www.nvidia.com/object/cuda_home_new.html
4. http://www.khronos.org/opencl/
5. http://www.nvidia.com/
6. Lehtihet R, El Oraiby W, Benmohammed M (2011) Improved fingerprint enhancement performance via GPU programming. Image Processing and Communications Challenges 3. Springer Berlin Heidelberg 102: 13-21.
7. Xiong Z, Chi W, Lu K, Wang X, Li G (2012) GPU acceleration of saliency detection algorithm. 11th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, DCABES, pp. 48-51.
8. Awad I (2013) Fingerprint local invariant feature extraction on GPU with CUDA. Informatica (Slovenia) 37: 279-284.
9. Gutierrez PD, Lastra M, Herrera F, Benitez JM (2014) A high performance fingerprint matching system for large databases based on GPU. IEEE Transactions on Information Forensics and Security 9(1): 62-71.
10. Peralta D, Triguero I, Sanchez-Reillo R, Herrera F, Benitez JM (2014) Fast fingerprint identification for large databases. Pattern Recognition 47(2): 588-602.
11. Cappelli R, Ferrara M, Maltoni D (2010) Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(12): 2128-2141.
12. FVC-onGoing: on-line evaluation of fingerprint recognition algorithms.
13. Lastra M, Carabaño J, Gutiérrez PD, Benítez JM, Herrera F (2015) Fast fingerprint identification using GPUs. Information Sciences 301: 195-214.
14. Cappelli R, Ferrara M, Maltoni D (2015) Large-scale fingerprint identification on GPU. Information Sciences 306(C): 1-20.
15. Fons M, Fons F, Cantó E (2012) Biometrics-based consumer applications driven by reconfigurable hardware architectures. Future Generation Computer Systems 28(1): 268-286.
16. Fons M, Fons F, Cantó E, López M (2012) FPGA-based personal authentication using fingerprints. Journal of Signal Processing Systems 66(2): 153-189.
17. Jiang RM, Crookes D (2008) FPGA-based minutia matching for biometric fingerprint image database retrieval. Journal of Real-Time Image Processing 3(3): 177-182.
18. Wu C, Liu Z, Feng C (2015) Palmprint minutia point matching algorithm and GPU application. Proceedings of Science 12: 1-9.
19. Refaie MN, Salman AA, Ahmad I (2011) Hybrid parallel approach based on wavelet transformation and principle component analysis for solving face recognition problem. International Conference on ICT and Knowledge Engineering pp. 72-79.
20. Abate F, Nappi M, Ricciardi S, Sabatino G (2007) GPU accelerated 3D face registration/recognition. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4642: 938-947.
21. Devani U, Nikam VB, Meshram BB (2014) Super-fast parallel eigenface implementation on GPU for face recognition. Proceedings of 2014 3rd International Conference on Parallel, Distributed and Grid Computing, PDGC 2014, India, pp. 130-136.
22. Kawale MR, Bhadke Y, Inamdar V (2014) Parallel implementation of eigenface on CUDA. 2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014, India.
23. Woo Y, Yi C, Yi Y (2013) Fast PCA-based face recognition on GPUs. Proceedings- IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP, Canada, pp. 2659-2663.
24. Vandal NA, Savvides M (2010) CUDA accelerated Iris template matching on Graphics Processing Units (GPUs). IEEE 4th International Conference on Biometrics: Theory, Applications and Systems, BTAS 2010, USA.
25. Sakr FZ, Taher M, Wahba AM (2011) High performance iris recognition system on GPU. Proceedings - ICCES'2011: 2011 International Conference on Computer Engineering and Systems, Egypt, pp. 237-242.
26. Broussard RP, Ives RW (2011) Using a commercial graphical processing unit and the CUDA programming language to accelerate scientific image processing applications. Proceedings of SPIE- The International Society for Optical Engineering 7872: 1-9.
27. Rakvic R, Broussard R, Ngo H (2016) Energy Efficient Iris Recognition with Graphics Processing Units. IEEE Access 4: 2831-2839.
28. Liu C, Petroski B, Cordone G, Torres G, Schuckers S (2015) Iris matching algorithm on many-core platforms. 2015 IEEE International Symposium on Technologies for Homeland Security, HST 2015, USA.