A GPU-based Evolution Strategy for Optic Disk Detection in Retinal Images[*]

Germán Sanchez-Torres[**]
Guillermo González-Calderón[***]

Received: 16/01/2015 • Accepted: 14/07/2016
DOI: 10.22395/rium.v15n29a11

Abstract

Parallel processing using graphic processing units (gpus) has attracted much research interest in recent years. Parallel computation can be applied to evolution strategy (es) for processing individuals in a population, but evolutionary strategies are time-consuming when used to solve large computational problems or complex fitness functions. In this paper, we describe the implementation of an improved es for optic disk detection in retinal images using the Compute Unified Device Architecture (cuda) environment. In the experimental results, we show that the computational time for optic disk detection task has a speedup factor of 5x and 7x compared to the implementation on a mainstream cpu.

Key words: Compute Unified Device Architecture, optic disk, evolutionary strategy, retinal images.

[*] Research article, partial result of the research project entitled "Automatic generation of injury mapping in retinographies", August 2012 to December 2013. Funded by the internal call for research projects Fonciencias 2012, Vice Dean for Research, University of Magdalena, Colombia.

[**] B.S., M.Sc., Doctor of Engineering. Assistant Professor, Faculty of Engineering, Universidad del Magdalena. Research and Development Group on New Information Technologies and Organizations. Santa Marta - Magdalena, Colombia. Carrera 32 N.° 22-08, Ed. Docente, Cub. 3D102, Phone: (57) 5- 4217940 Ext. 1138, E-mail: gsanchez@unimagdalena.edu.co

[***] B. S, M.Sc., Doctor of Engineering, National University of Colombia. Medellín, Antioquia, Colombia. Cl. 59a #63-20, Phone. (57) 4-25 51 16. Fax. (574) 4-25 51 16, E-mail: ggonzal@unal.edu.co

Estrategia evolutiva basada en GPU para la detección del disco óptico en imágenes de retina

Resumen
La ejecución paralela de aplicaciones usando unidades de procesamiento gráfico (gpu) ha ganado gran interés en la comunidad académica en los años recientes. La computación paralela puede ser aplicada a las estrategias evolutivas para procesar individuos dentro de una población, sin embargo, las estrategias evolutivas se caracterizan por un significativo consumo de recursos computacionales al resolver problemas de gran tamaño o aquellos que se modelan mediante funciones de aptitud complejas. Este artículo describe la implementación de una estrategia evolutiva para la detección del disco óptico en imágenes de retina usando Compute Unified Device Architecture (cuda). Los resultados experimentales muestran que el tiempo de ejecución para la detección del disco óptico logra una aceleración de 5 a 7 veces, comparado con la ejecución secuencial en una cpu convencional.

Palabras clave: gpu, disco óptico, estrategias evolutivas, imágenes de retina.

INTRODUCTION

Evolutionary strategies belong to nature-based models for stochastic optimization, namely Evolutionary Computation (EC). EC is an effective method for solving many difficult optimization problems. However, when handling with large scale and complex tasks computation time can be considerable. EC generally entails large computational costs because it usually evaluates all solution candidates in a population for every generation [1].

Nature-inspired operations, such as selection, reproduction, and replacement, are used to explore a specific solution space to evolve the population toward the best solution automatically [2]. The nature of EC algorithms, such as ES, is inherently parallel since population members are typically subjected to the same operations. But parallel computing facilities were not widely available and affordable in the past, so the majority of existing EC-based solutions were designed and implemented in a sequential way.

Since during the past two decades hardware availability was limited, parallel processing such as cluster computing was applied in order to reduce overall computing time in EC models. One of the main focuses of research in parallel evolutionary computation is the parallelization of one or more of the typical steps of an EC solution, such as the design of a parallel fitness function, parallel estimation of individual fitness function, and individual generation in parallel. A parallel fitness function evaluator reduces computing time because the calculation of fitness function is the most time consuming part of EC solutions. To parallelize the fitness function estimation between individuals is possible because computation for each individual is independent from the others. In a similar way, the crossover, mutation, and selection schemes can be applied in a parallel way to an initial set of individuals in order to process a generation.

In [3], three methods are described for the parallel implementation of a specific model of EC, namely Genetic Algorithms (GA). Since the main difference between GA and ES is related to individual encoding, the three methods can be extended to ES; these are the master slave model, the coarse grained model and the fine grained model, where individuals run on different processors at the same time, which allows for a fast calculation of the fitness function, but requires specialized hardware with many cores to efficiently take advantage of the model. The great level of parallelism of this model made it suitable for SIMD (single instruction, multiple data) systems [4] like CUDA [5]. Additionally, hybrid models can be used to maximize the parallel facilities for problem solving.

CUDA is the hardware and software architecture that enables NVIDIA GPUS to execute program calls in parallel kernels. The natural parallelism of computation on

GPUs is expressed by a few compiler directives added to the C programming language [6]University of Tennessee and Oak Ridge National Laboratory CUDA is a computing architecture designed to facilitate the development of parallel programs. In conjunction with a comprehensive software platform, the CUDA Architecture enables programmers to draw on the immense power of graphics processing units (GPUs. An important consideration in software development on the CUDA architecture is related to the correct use of multiple memory levels in order to reduce latency on data access. Specific hardware capabilities must also be considered, since each device type has different configurations like SM quantity, amount of memory, maximum thread number by SP, maximum grid and block dimension capability, threads grouping for execution or wraps, and control divergence [6-7] that must be taken into account for correct algorithm execution.

Some research in the EC parallel model design using CUDA is presented in [8]; where CUDA was used on a G80 GPU to efficiently interpret several GP programs in parallel obtaining speedups ranging from 8x to 80x. Many techniques have been proposed in different knowledge areas like medical image processing [9]. We are interested in the area for diabetic retinopathy detection.

Diabetic retinopathy is a progressive disease, its diagnosis is based on some clinical abnormalities which are difficult to detect because it is asymptomatic until advanced stages of disease development [10]. Without adequate treatment, it can evolve into a more complicated condition called Macular Edema [11]. Typically, the procedures used for retinal analysis include the identification of ocular physiology elements, like blood vessels, optic disc, and fovea [12-13]. These parts must be identified so that they are not taken into account by the algorithms designed to detect the presence of disease signs such as aneurisms, bleeding or exudates.

We designed an ES-based search method to determine the coordinates of the optic disk (OD) in retinal images. The search process is guided through a fitness function that joins the two most representative physiological characteristics of the OD region: high brightness areas and high presence of blood vessels.

In this paper we present the design of a GPU implementation of ES addressing optical disk detection and segmentation using CUDA technology in order to speed up program execution to address the computational limitations of a previously reported sequential design [14].

1. METHODOLOGY

1.1 Optic disk detection using evolution strategy

Localization methods are based on two physiological characteristics of the optic disk. First, due to its organic composition, the region in which the OD is located corresponds to the brightest components [15]. However, a major limitation related to this fact is the absence of standardization of technical characteristics for retinal image acquisition. As a result of uncontrolled camera parameter variations, the OD region is not necessarily the brightest area of the image.

Another physiological characteristic is that all vessels and sanguine branches originate in the optic disk. It is known that these branches follow parabolic trajectories that converge to the OD. These characteristics have been used to design techniques for optic disk detection [16]. Other works, such as [17], apply a preprocessing approach for initial treatment in order to enhance contrast by using the HSI (Hue, Saturation, and Intensity) channel; optic disk localization is achieved through intensity variance inside the OD. Variance is caused by bright pixels and dark pixels from vessels. Another example of vessel guidance for OD localization is a technique called fuzzy convergence [18], which consists in finding the point of convergence of the blood vessels.

Optic disc localization in retinal images is not a trivial procedure due to color space variations caused by the ethnicity of the individual, features of the acquisition device, and optical disturbances introduced by the presence of clinical abnormalities. For all of the above, selecting the brightest region of a retinal image is not a guaranteed method for detecting the location of the optical disk.

The evolution strategy employed was ES ($\mu + \lambda$), where a set of μ parents are used to create a λ offspring in each generation. In this strategy, the μ parents and λ offspring constitute a whole individual set called population. After each iteration, the best μ individuals are selected by means of a fitness function, which is a user-defined function based on a specific heuristic in the problem's context. Subsequently, crossover and mutation operators are applied to these individuals generating λ offspring in order to form a new individual set to apply the whole process again.

From a general point of view, search techniques using evolutionary strategies follow a workflow based on selecting and generating individuals through validation of a fitness function, similarly to Genetic Algorithms (GA). One of the main differences between Evolution Strategies and Genetic Algorithms is that the latter require the definition of functions for encoding and decoding the internal structures of individuals. Due to the circular nature of the target pattern to be detected, each individual is represented through a circle. Each circle represents an individual of radius r located at coordinates (x, y) (Figure 1).

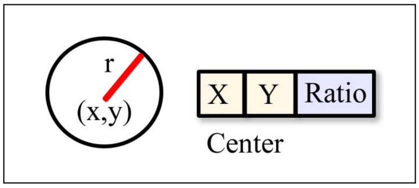

Figure 1. Representation of an individual in ES.
Source: authors

The objective is to determine the circle of radius r which maximizes the number of bright pixels and contains the largest number of edges of the blood vessel network. Thus, the fitness function is defined as:

$$fitness_i = f_b(x,y,r) + f_e(x,y,r) \tag{1}$$

where, $f_b(x,y,r)$ constitutes a percentage measure of the amount of bright pixels within the circle represented by (x,y,r). This measure is estimated in relation to the total number of bright pixels and calculated using the 10 % brightest pixel set. Similarly, $f_e(x,y,r)$ is a percentage measure of the number of edges inside the circle in relation to the total number of pixels obtained from an edge image.

The crossing pattern used corresponds to the general recombination scheme in which the offspring values are estimated as the sum of the values of parents, thus:

$$C = c_1 + \eta(c_1 + c_2) \tag{2}$$

where c_1 and c_2 are the values of the parent's characteristics and ç is a random number normally distributed.

The mutation operator is applied altering each individual value by adding a quantity, which is based on normally distributed random numbers:

$$C = C + N(0,\sigma) \tag{3}$$

where C represents each feature referred to the center of the circle.

1.2 CUDA model implementation

In CUDA, parallel threads run in an organized structure defined by the compiler process, and determined by the programmer. This structure is made of thread blocks and a grid of thread blocks. A thread block is a set of threads running concurrently, sharing memory and synchronization facilities, while a grid is an array of thread blocks executing the same subprogram called kernel. Figure 2 shows the general architecture scheme of a CUDA device. It has multiple streaming multiprocessors (SM) and each has multiple streaming processors (SP) [5].

Each thread has its Local Memory space, while threads in a block can share data using a common Shared Memory space and each thread block group in a grid can access data using a Global Memory space (Figure 3).

The three memory levels exist inside GPU devices; however, since parallel execution of a program consists of one or more phases executed on CPU and others on GPU, a general level of memory is the CPU memory [19]. Thus, memory considerations include the data transfer between CPU and global memory on GPU [7].

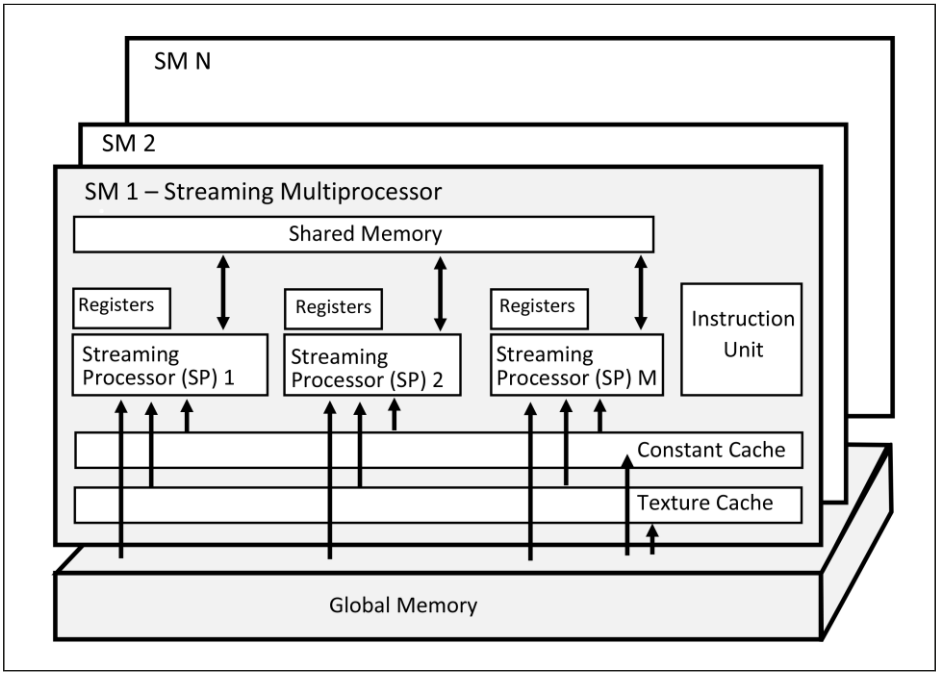
SM N
SM 2
SM 1 – Streaming Multiprocessor
Shared Memory
Registers
Registers
Registers
Streaming Processor (SP) 1
Streaming Processor (SP) 2
Streaming Processor (SP) M
Instruction Unit
Constant Cache
Texture Cache
Global Memory

Figure 2. CUDA architecture description
Source: [5]

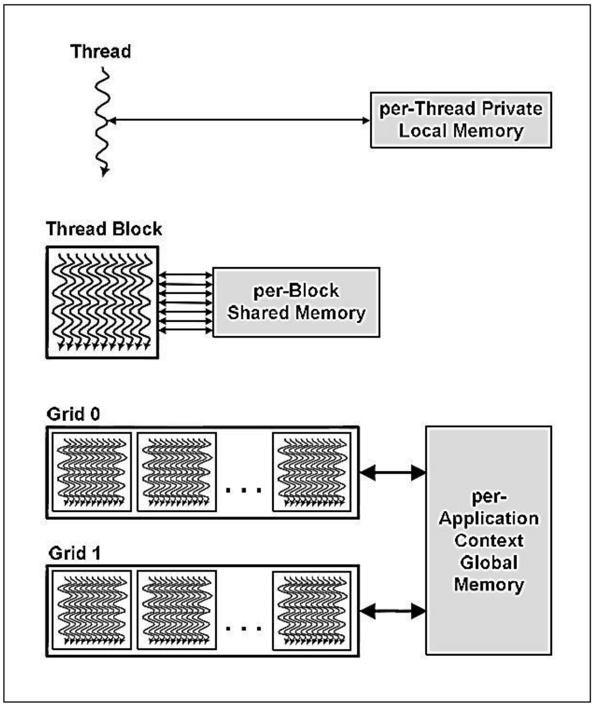
Thread
per-Thread Private Local Memory
Thread Block
per-Block Shared Memory
Grid 0
Grid 1
per-Application Context Global Memory

Figure 3. CUDA memory hierarchy of the thread, block and grids
Source: [5]

Different schemes for GA parallelization have been proposed: Master/Slave, Synchronous Island, Asynchronous Island and Cellular scheme [20], we extended these to the Evolution Strategy. A unique solution to design an optimal parallel GA does not exist, it depends on specific characteristics of the problem in question.

In the Master/Slave model the tasks are divided between CPU and GPU execution; thus, the whole population of a GA is centralized on CPU memory. In the Island model, the population is divided into several sub populations, called islands, which evolve independently from each other; therefore, all typical steps of GA execution are carried out for each island, with one additional step being added to the algorithm, i.e. migration. Migration consists of transferring the information of the best individual after some iterations so that the whole algorithm evolves to the global optimum. Specifically, the way migration takes place differentiates between the synchronous and asynchronous island models. In the synchronous model, migration takes place after a specific number of generations for all islands; this means that all islands must be synchronized for the global process to continue. By contrast, in the asynchronous model each island does the migration after a fixed number of generations, regardless of whether the other islands have reached the same number of generations. Both the synchronous and asynchronous island models are adequate for GA execution in a multicore architecture. Finally, in the cellular scheme for GA implementation, individual evolution is executed on separate devices. Thus, each computing device is viewed as a node of a connected topology, and each node processes a different individual. To apply the crossover and mutation operators connected, neighbors are taken into account. This scheme is called a *fine grained scheme* while the other models are called *coarse grained schemes*.

2. RESULTS

Experiments were conducted on a PC equipped with an Intel 2.2 GHz CPU and a NVidia GeForce GTX 550 Ti GPU with 1024 MB GDDR5 of global memory and 192 CUDA cores. The software was CUDA driver and SDK with version 5.0 using C++ language. The set of retinal images used was composed of 491 images taken from the DIARETDB (Diabetic Retinopathy Database and Evaluation Protocol) public repositories [21].

2.1 Time consuming analysis and selection of execution model

In order to select a parallel model, we analyzed the computational cost of sequential implementation of ES described in the previous section in terms of its time consumption. Figure 4 shows the perceptual average of time spent in crossover, mutation, evaluation and selection stages for a set of 200 algorithm executions. The evaluation stage takes about 46 % of the total execution time because algorithms must operate on two distinct images: $f_b(x,y,r)$ is estimated from a normalized intensity image and $f_e(x,y,r)$ is estimated from a border image in order to count the number of border pixels which represent blood vessel presence.

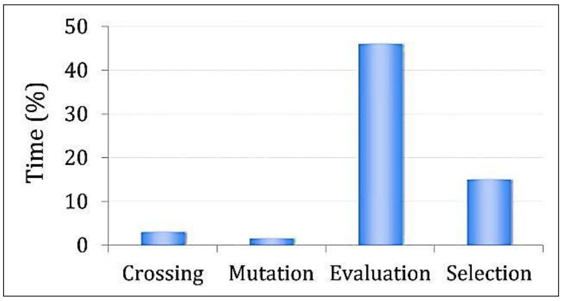

Figure 4. Perceptual time consuming cost of sequential execution of the main stages of the proposed ES.
Source: authors

The second major cost was found in the selection stage. This includes the sorting operation to select the best λ individual in each generation. Consequently, we decided to optimize the execution time in the most time consuming stage. Thus, the parallel ES was based on the Master/Slave scheme, in which the master process runs on a CPU where selection, crossover and mutation operations are applied. The Slave processes run on the CUDA cores and only perform the fitness value estimation in parallel (see Figure 5).

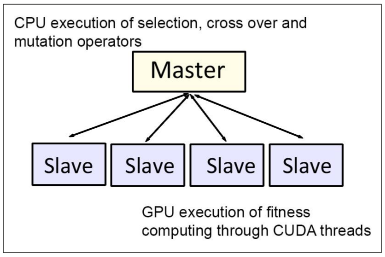
CPU execution of selection, cross over and mutation operators
Master
Slave
Slave
Slave
Slave
GPU execution of fitness computing through CUDA threads

Figure 5. General scheme of the proposed evolution strategy.
Source: authors

For the GPU based estimation of the fitness function values, we loaded the information of the two images in the global memory space on the device. These are the normalized image and border image representations. The normalized image was used to estimate the f_b factor, which was obtained from the original retinal image by applying a median filter using a window size of 3 % of the original image. The final pixel value was obtained by adding the green and red channel values in a 70 % and 30 % proportion respectively. This allowed us to increase the contrast of the bright regions while red zones were maintained. The border image was estimated by applying the classic Canny edge detection algorithm which has shown better results than other operators [22]. The f_e value is estimated from this image. Figure 6 shows examples of normalized and border images.

Additionally, the population information was also loaded to the device because it is necessary for the circle estimation used for counting bright and border pixels. Each CUDA thread is used to estimate one fitness value; thus, each thread reads one set of (x, y, z, r) values and estimates the f_b and f_e values.

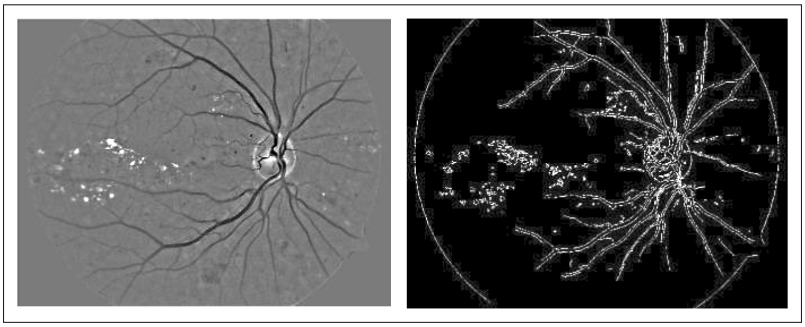

Figure 6. a) Normalized image and b) border image.
Source: authors

Initially, on the CPU side, the total bright and border pixels are estimated and passed to CUDA threads for perceptual estimation. On the GPU side, in order to speed up the bright and border pixel count process inside the circles, each i individual generates a set of k random position $\left\{\left(x_1^i, y_1^i, z_1^i\right), \cdots, \left(x_k^i, y_k^i, z_k^i\right)\right\}$ from the center, and lying inside the circle of its' radius length. This process is straightforward because it can be done by generating random numbers smaller than the length of the radius and adding to or subtracting from the coordinates of the center. The random number generation process is relevant for good performance of all types of EC algorithms, and in the parallel context, random number generation is not a trivial procedure due to the intrinsic sequential nature of the typical Linear Congruential Generators (LCG). This problem has been addressed in works such as [23]. CUDA 3.2 included a library for GPU accelerated random number generation routines, supporting Sobol quasi random and XORWOW pseudo random routines [5]. The whole scheme of ES execution is illustrated in figure 7.

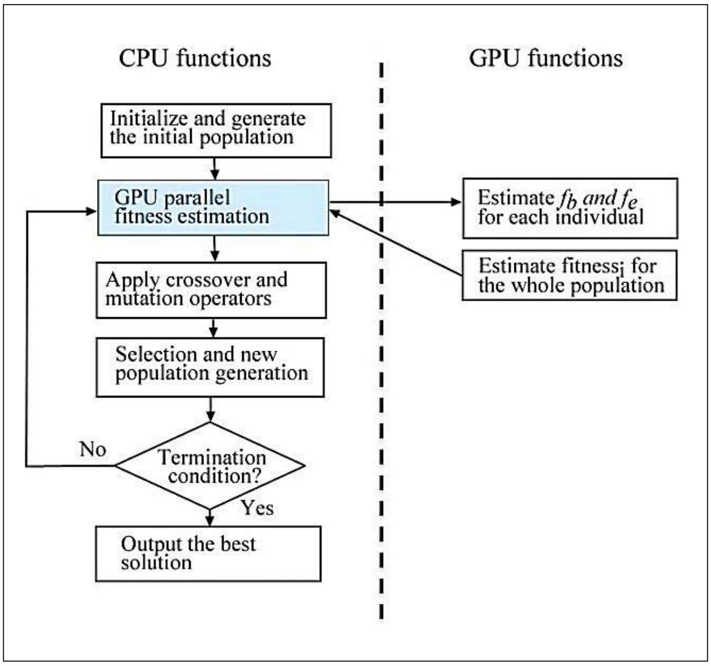
CPU functions
GPU functions
Initialize and generate the initial population
GPU parallel fitness estimation
Apply crossover and mutation operators
Selection and new population generation
Termination condition?
No
Yes
Output the best solution
Estimate f_b and f_e for each individual
Estimate fitness$_i$ for the whole population

Figure 7. Algorithm of ES executing on CPU and GPU.
Source: authors

2.2 Optic disk detection

Experimentally, we found that the bright pixels forming the optic disk were in the range of 15 % to 20 % of the brightest pixel set in the whole image. Figure 8 shows an example of this initial pixel selection.

Other initial tests were designed to determine the ES operating parameters. We analyzed the convergence of ES by estimating the distance from the center of the circumference of the best individual to a predefined and manually set point *pr* located in the center of the OD.

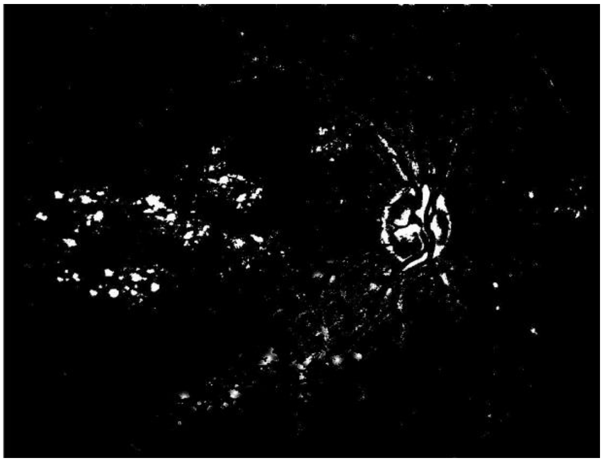

Figure 8. Initial selection of the brightest pixel set.
Source: authors

Figure 9 shows the average behavior, for a set of twenty runs, of the average distance between *pr* and the ES solution. After 60 iterations the average stabilized; the distance decreased after about 80 iterations.

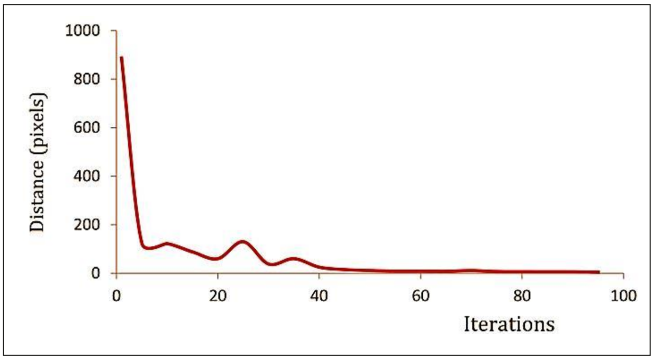

Figure 9. Average behavior of the ES best individual convergence of ES.
Source: authors

The initial radius size of the circumferences was experimentally set based on the fact that an average healthy optical disk can be represented by a circle of a 100 pixel radius in an image of 1,500 x 1,152 pixels. The EE parameterization was made according to table 1.

Table 1. Parameter values of proposed es

Parameter	Value
Mutation prob.	0.01
Crossover prob.	10
Initial radius	100
Random positions (k)	100
Max. generations	100

Source: authors

Figure 10 shows the typical fitness behavior of the best individual through 150 iterations. The fitness value tends to increase until 80 iterations. From then on the fitness value seems to reduce the improvement rate to just over 110 iterations.

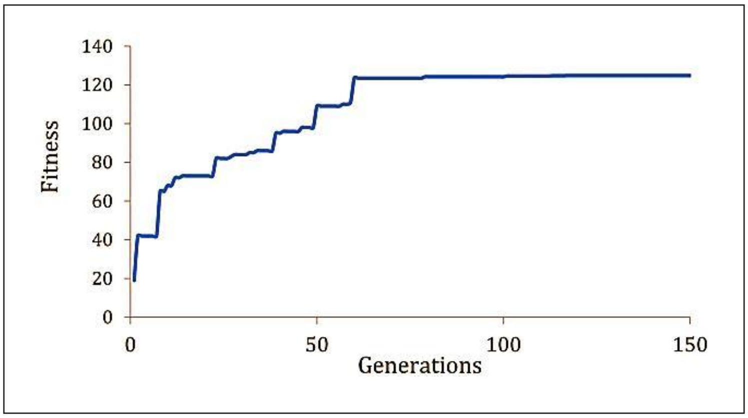

Figure 10. Results of a typical run of the ES.
Source: authors

Graphical results can be seen in figure 11. The method was applied to a 491 image set including images from healthy and diseased individuals. The main diseases found in the images are hard and soft exudates. The results were 469 images with correct optic disk localization and 22 with misplaced optic disk localization. We consider an acceptable optic disk localization as one that contains 60 % or more pixels of the real optic disk region.

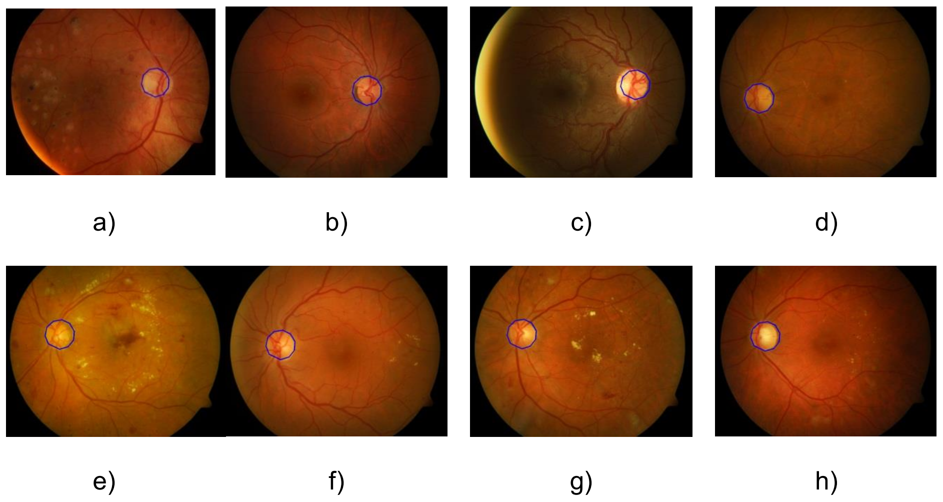

Figure 11. Graphical examples of ES solutions.
Source: authors

In general, we did not expect an improvement in the accuracy of optic disk localization with regard to the sequential model. Consequently, an accuracy comparison is irrelevant. The greatest effects are found on computation time when varying the parameter values of the algorithm.

In view of the searching nature of ES algorithms, a way to enhance the final result is to expand the population size, which permits to have more possible solutions and enlarge the search space. Figure 12a shows the average computation time used by the CPU and GPU executions when varying the population size from 100 to 1,000

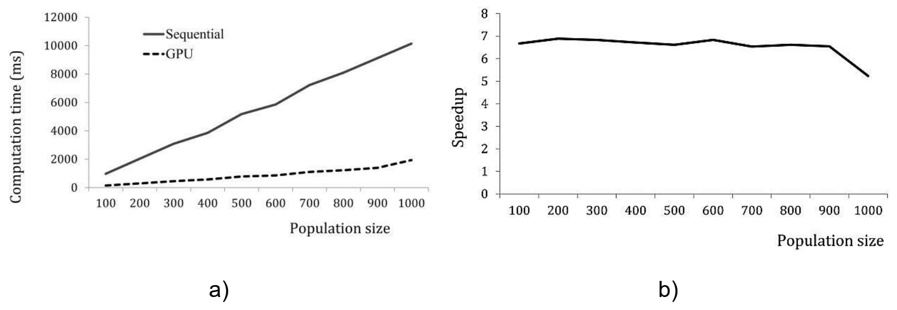

Figure 12. a) CPU and GPU computation time (ms) according to population size variations and b) GPU speedup behavior according to population size variations.
Source: authors

individuals. The GPU model is able to reduce the time used in a factor of *5x* to *7x* for some specific population values.

The speedup obtained is shown in figure 12b. In the 100 to 200 individual range the algorithm achieved a speedup of 7*x* approximately, and it decreased to 5*x* for a population size of 100 individuals.

3. DISCUSSION

The proposed model reduces the average computation time 6.5 times when compared to the sequential method. The largest increases were achieved when the population size was smaller. This is because of the hardware on which the algorithm was executed. The CUDA technology uses the blocks of threads concept. Such blocks are assigned to each Streaming Multiprocessor (SM). Inside blocks, the threads are divided into warps or 32 thread sets, which are executed simultaneously. Therefore, if the population size can be exactly mapped to warps and SM, the fitness computation of the whole population is carried out in parallel. However, when the population size grows and the subdivision is not exact, other factors impact the required computational time. One of the most important factors is *divergence control*.

Divergence control takes place in the presence of conditionals causing different paths for threads inside warps. The different control paths taken by threads in a warp are traversed one at a time until there are no more paths. Avoiding divergence control is a programmer's task and it is specific of domain problem features. This is a factor that may generate speedup variations when a whole population is processed, as shown in figure 12b.

4. CONCLUSION

ES based on bright and vessel border pixels for optic disk localization showed an accuracy of 96 %. Computational time decreased when using a GPU based execution model. The proposed model reaches a speedup of 6 *x* compared to a sequential model.

GPU inclusion in medical image processing constitutes an alternative in order to reduce the high computational costs associated with image processing techniques.

The main limitation found when processing retinal images for optic disk localization is related to bright pixels when the disease is at an advanced stage because high presence of exudates, common at such stages, increases the proportion of bright pixels. Moreover, retinal images with low contrast make correct localization more difficult. As future work, we plan to expand the proposed method to decrease the effect of large exudate regions on localization accuracy.

REFERENCES

[1] O. Kramer, “Evolution Strategies”, in *A Brief Introduction to Continuous Evolutionary Optimization*, Berlin: Springer International Publishing, 2014, pp. 15-26.

[2] N. Hansen, D. V. Arnold & A. Auger, “Evolution Strategies”, in *Springer Handbook of Computational Intelligence*, J. Kacprzyk & W. Pedrycz, Eds. Berlin: Springer, 2015, pp. 871-898.

[3] G. Zeng & C. Ding, “An Analysis on Parallel Genetic Algorithm,” *Computer Engineering*, vol. 27, no. 9, pp. 53-55, 2001.

[4] Y. Ke, Y. Li & D. Li, “Image Matching Using Genetic Algorithm on GPU”, in *2011 International Conference on Control, Automation and Systems Engineering (CASE)*, Singapore, Jul. 30-31, 2011, pp. 1-4.

[5] *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide. Version 1.0.* NVIDIA Corporation, California, USA, 2007. Available: http://developer.download.nvidia.com/compute/cuda/1.0/NVIDIA_CUDA_Programming_Guide_1.0.pdf

[6] J. Sanders & E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, 1st ed. Michigan, USA: Addison-Wesley Professional, 2010.

[7] D. Kirk & W. Hwu, *Programming Massively Parallel Processors*, 2nd ed. Massachussetts, USA: Elsevier, 2012.

[8] D. Robilliard, V. Marion-Poty & C. Fonlupt, “Population Parallel GP on the G80 GPU”, in *Genetic Programming*, M. O’Neill, L. Vanneschi, S. Gustafson, A. I. E. Alcázar, I. D. Falco, A. D. Cioppa & E. Tarantino, Eds. Berlin: Springer 2008, pp. 98-109.

[9] G. Chen, D. Xu, H. Hu, Y. Liu & R. Chen, “The Application of CUDA Technology in Biomedical Image Processing”, in *Emerging Research in Artificial Intelligence and Computational Intelligence*, J. Lei, F. L. Wang, H. Deng & D. Miao, Eds. Berlin: Springer, 2012, pp. 378-385.

[10] M. A. Khan & A. Juhn, “Diabetic Retinopathy,” in *Optical Coherence Tomography*, A. Girach & R. C. Sergott, Eds. Berlin: Springer International Publishing, 2016, pp. 29-42.

[11] L. Giancardo et al., “Exudate-based diabetic macular edema detection in fundus images using publicly available datasets”, *Medical Image Analysis*, vol. 16, no. 1, pp. 216-226, Jan. 2012.

[12] M. Krause, R. M. Alles, B. Burgeth & J. Weickert, “Fast retinal vessel analysis”, *J Real-Time Image Proc*, vol. 11, no. 2, pp. 413-422, Apr. 2016.

[13] O. S. Soliman, J. Platoš, A. E. Hassanien & V. Snášel, “Automatic Localization and Boundary Detection of Retina in Images Using Basic Image Processing Filters”, in *Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011*, M. Kudělka, J. Pokorný, V. Snášel & A. Abraham, Eds. Berlin: Springer, 2013, pp. 169-182.

[14] G. Sánchez Torres, A. Espinosa Bedoya & Y. Fernando Ceballos, “Detección del disco óptico en retinografías mediante una estrategia evolutiva (μ+λ),” *Revista EIA*, vol. 11, no. 21, pp. 55-66, Jun. 2014. Available: http://www.scielo.org.co/pdf/eia/n21/n21a05.pdf

[15] M. D. Abramoff & M. Niemeijer, "The automatic detection of the optic disc location in retinal images using optic disc location regression", in *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS '06*, New York, USA, Aug. 31 - Sep. 3, 2006, pp. 4432-4435.

[16] S. Sb & V. Singh, "Automatic Detection of Diabetic Retinopathy in Non-dilated RGB Retinal Fundus Images", *International Journal of Computer Applications*, vol. 47, no. 19, pp. 26-32, Jun. 2012.

[17] C. Sinthanayothin, J. Boyce, H. Cook & T. Williamson, "Automated localisation of the optic disc, fovea, and retinal blood vessels from digital colour fundus images", *Br J Ophthalmol*, vol. 83, no. 8, pp. 902-910, Aug. 1999.

[18] A. Hoover & M. Goldbaum, "Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels", *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 951-958, Aug. 2003.

[19] C. Trujillo & J. García-Sucerquia, "Graphics Processing Units: More Than the Pathway to Realistic Video-Games", *Dyna*, vol. 78, no. 168, pp. 164-172, 2011.

[20] L. Zheng, Y. Lu, M. Ding, Y. Shen, M. Guoz & S. Guo, "Architecture-based Performance Evaluation of Genetic Algorithms on Multi/Many-core Systems", in *2011 IEEE 14th International Conference on Computational Science and Engineering (CSE)*, Dalian, China, Aug. 24-26, 2011, pp. 321-334.

[21] V. Kalesnykiene, J. Kamarainen, R. Voutilainen, J. Pietilä, H. Kälviäinen & H. Uusitalo, *DIARETDB1 diabetic retinopathy database and evaluation protocol*. Machine Vision and Pattern Recognition Laboratory Lappeenranta University of Technology, 2012.

[22] V. G. Narendra & K. S. Hareesh, "Study and comparison of various image edge detection techniques used in quality inspection and evaluation of agricultural and food products by computer vision", *Int J Agric & Biol Eng*, vol. 4, no. 2, pp. 83-90, 2011.

[23] W. B. Langdon, "A Fast High Quality Pseudo Random Number Generator for nVidia CUDA", in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, New York, NY, USA, 2009, pp. 2511-2514.