Computational Efficiency Optimization of Optic Disc Detection in Fundus Image

Sofien Ben Sayadia, Yaroub Elloumi, Rostom Kachouri, Mohamed Hedi Bedoui

Computational Efficiency Optimization of Optic Disc Detection in Fundus Image

Sofien BEN SAYADIA [1,2,3], Yaroub ELLOUMI [1,2,3], Rostom Kachouri [2], Mohamed Hedi BEDOUI [1]

[1] Medical Technology and Image Processing Laboratory, Faculty of medicine, University of Monastir, Tunisia.
[2] LIGM, Univ Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée.
[3] Université de Sousse, ISITCom, 4011, Hammam Sousse, Tunisie.

sofien.bensayadia@esiee.fr

Abstract. The OD detection is a mainly step in many methods for ophthalmic diseases diagnosis. The work described in [7] proposes a performance optic disk detection approach based on blood vessel tracking and optic disk contrast. However, the method is characterized by a higher execution time which is about 10 s in STARE DB images. Moreover, the execution time increases proportionally with the fundus image resolution. This computational performance is a limiting factor to employ the method in diagnosis systems of ophthalmic diseases.
This paper aims to optimize the method processing in order to enhance the computational performance. The first contribution consists of optimizing repetitive steps with the aim of reducing times. Thereafter, all processing steps are implemented in GPU architectures. The experimental results indicate that each one of the contributions insures enhancing computational performance with speedup equal to 1.7 and 2.5, respectively. The implementation with combined contributions leads to a speedup equal to 8.6 which leads to an execution time about 1 second.

Keywords: fundus image; Optic Disc localization, Parallel algorithms, Graphics Processing unit (GPU), CUDA, Real-time GPU implementation.

1 Introduction

The Optic Disk (OD) is a main retinal anatomic structure in fundus image. The OD detection is a critical step in many diagnostic systems for ophthalmic diseases. In the case of diabetic retinopathy (DR), the OD has the same color and contrast than Hard Exudates (HEs). Therefore, several works [11,2,3,4,5] proceed to detect and remove OD before segment HEs. Moreover, the works described in [6,18,8] to detect the neovascularized blood vessels in the OD in order to deduce the proliferative DR. In the case of the Glaucoma, several works aim to extract the OD and the Cup Disc where the size ratio indicate the presence of the disease [9,10,13] and the pathology level [24]. Other works detect the Glaucoma by figuring out the position of the cup disc

with respect to the OD. The Age-related Macular Degeneration (AMD) is always detected through the drusens segmentation which is located in the macula and has the same contrast than the OD. Therefore, the works described in [32] aim to detect all shape having a higher contrast, and then eliminate the OD in order to detect the drusens. Moreover, several approaches are proposed in the objective of extracting the macula [16,14], the fovea [15] and the blood vessels [17], which OD location is an important processing step.

A significant number of OD localization methods are proposed in the literature. Recent OD localization approaches offer both high and close detection performance. Those methods proceed to detect the OD based on their retinal characteristics such as the brightness, the contrast and circular shape. The algorithm described in [25] employs the Radon Transform (RT) to localize OD based on its brightness and roundness. The RT is applied with several angles to each sub-window in the objective of detecting OD circular shape. Hashim et al. [26] apply a binary mask on the intensity channel to exclude the background pixels. Then, morphological operators and contrast enhancement techniques (Gamma transformations) are used in conjunction with the difference of the Gaussian filter (DOG) to obtain the OD border. In the work of Giraddi et al. [27], a thresholding is employed to eliminate false positive based on the OD brightness and roundness shape. Then, the OD segmentation is performed using the vector field gradient (GVF snake). These methods present high success rates in normal images. However, the detection provides inaccurate results due to the presence lesions having the same brightness or size than the OD.

Others OD detection methods are based on vessel tracking. Foracchia et al. [29] used a parametric geometric model (parabolic path) to describe the typical direction of the vessel structure. In the work proposed by Zhang et al. [28], the density, compactness and uniformity of blood vessels are formulated to find the OD coordinates. Then, the matched filter is applied in various dimensions in order to provide candidate location. Those approaches tend to be the most Robust in OD appearance change. However, they can provide a wrong OD detection if the vascular network is partially extracted.

Some others OD detection methods employ the OD characteristics, and exploit the location and orientation of vessels. For example, Youssif et al. [30] uses the directional pattern of retinal blood vessels for the OD detection. Their method involves normalizing contrast and luminosity. Xiong and Li [31] have proposed a method for locating the OD center by extracting a variety of features including vertical and horizontal vessel intensity and the size of the bright object. Soares et al.[7] proposes an algorithm based on the cumulative sums of successive subdivisions and the vessel enhancement. The next step consists at following vessel convergence to locate the OD. These approaches tend to be the most effective and reliable, even in incomplete appearance and change of OD, and in incomplete construction of the vascular structure. The work proposed in [7] achieves an OD localization accuracy of 99:15%. This performance is provided using eight public datasets including the STARE and DRIVE ones.

However, the method is characterized by a higher execution time which is about 10 s in STARE DB images where the resolution is equal to (700 * 605). Moreover, the execution time increases proportionally with the fundus image resolution. As an ex-

ample, the current retinographs TRC-NW 7 SF[12] provides fundus images with resolution equal to (3008 x 2000) which is 14 times greater than a STARE dataset image, and hence a similar rise on execution time. This computational performance is a limiting factor to employ the method in diagnosis systems of ophthalmic diseases.
This paper proposes optmize and parallelize the processing method proposed in [7] with the aim of reducing the time of execution. The article is organized as follows. In Section II, we describe the approach of OD detection. In section III, we analyze processing times and complexity in terms of approximate number of operations. Then, we describe our proposed contributions for the acceleration of the OD detection algorithm. The evaluation of proposed contributions using different retinal image databases is done in Section V, followed by Discussion and conclusion in last section.

2 Optic Disc Detection method [7]

This section presents a description of the method proposed by Soares et al [7] for detecting OD in the fundus image. The main idea is based on identifying the concentration and the convergence of the main vessels in order to detect the OD location. This method is composed by successive processing blocks as indicated in the flowchart in Fig.1 and which are described to the following sessions.

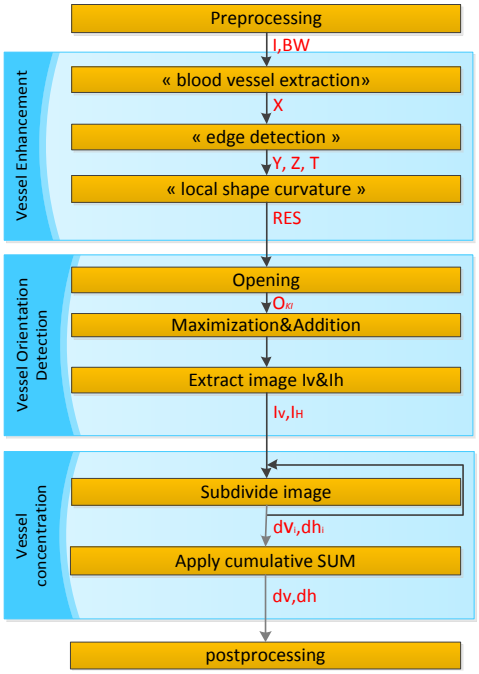


Fig. 1. Method proposed by Soares et al [7]

2.1 Preprocessing:

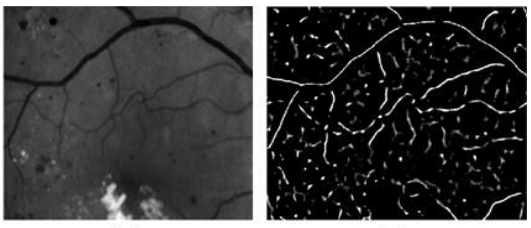
The first processing block entitled "preporcessing" starts by resizing the green component of the image to a resolution of (900 * 900) using a bicubic interpolation [7], in the case where the resolution is greater than (900 * 900). Then, the background is separated from the fundus image using a binary mask applied to the red channel where the threshold is equal to 28 whose result is saved in the BW image. The next step eliminates the noise by applying the "Gaussian Blur" filter with a rectangular structure of 13 × 13 pixels and $\sigma = 4$, the result is saved in image I.

2.2 Vessel Enhancement

The main objective of this block consists at extracting the main vessels from the retina vascular network based on their contrast and thickness. The first step, called "bloodvessel extraction", aims to reconstruct the thicker vesselsthat cross the OD.Thus, the Laplace and gradient filter are applied separately to the Iimage. Next, thevascular network is constructed by calculating the difference between the square absolute value of the gradient image and the Laplacian image.The result is saved in the image X (M * N) as shown in fig.2 (b).

The provided images are characterized by a higher noisy which avoid distinguish between vascular structures and non-vascular ones. Therefore, the Hessian matrix [7] is employed in order to and then removing the thin vessels, whose treatments are respectively entitled «edgedetection» and «Local Shape Curvature» («LSC»).

The second step, called “edge detection”, consists of describing an edge detector by using the second-order derivatives. The first-order derivatives are calculated by applying a "Sobel" filter three times on the image I, using three matrices with derivative orders respectively in the x direction, the y direction, and the x and y directions. Then, each "Sobel" filter is followed by a Gaussian filter for calculating the second-order derivatives. The results are recorded respectively in the Y, Z and T images.



(a) (b)

Fig. 2. Extract blood vessels networks.(a) Retinal imageI(M*N). (b)Image X(M*N).

The third step, called “Local Shape Curvature (LSC)” leads to distinguish vascular structures from non-vascular structures by minimizing the impact of false vessels and avoiding lesions such as micro-aneurysms and exudates. In fact, the background pixels are characterized by a small magnitude of the derivatives (the eigenvalues) relative to the values present in the sets of the vessel pixels. Therefore, the spatial derivative

matrix H is constructed for each point p of index i, j such that i∈ [0, N-1], j∈ [0, M - 1], as indicated in equation (1).

$$H = \begin{pmatrix} Y(i,j) & T(i,j) \\ T(i,j) & Z(i,j) \end{pmatrix} \quad (1)$$

Then, the eigenvalues λ1 and λ2, are calculated by solving the characteristic equation of the Hessian. These values are essential to distinguish, respectively, the minimum and maximum principal curvatures. The author indicates that the minimal eigenvalue λ1 represents a low contrast corresponding to the pixels belonging to the regions of the blood drops or the microaneurysms.However, the maximum Eigenvalue λ2 represents the points of interest corresponding to the pixels belonging to the main vessels that cross the DO [7].
For this, at each point p, the minimum eigenvalue is removed from the image of the vascular network X as presented in equation (2). The result is saved in RES images (Fig.3 (b)).

$$RES = X - min(X, \lambda 1) \quad (2)$$

Then, to maximize the impact of the main vessels of the vascular network, each pixel of the RES image is multiplied by the corresponding maximum eigenvalue (Fig.3 (c)).

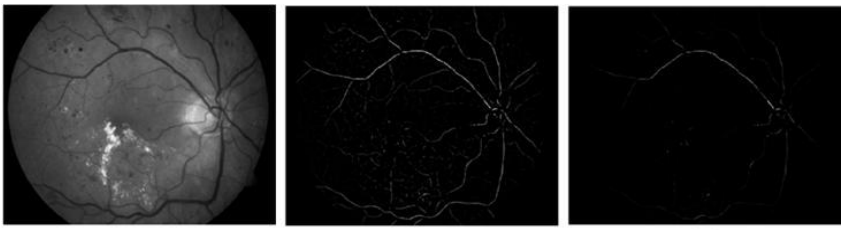


Fig. 3. Extract LSC:(a) Retinal image; (b) thin vessels removed;(c) Maximize the impact of principal vessels.

2.3 Vessel Orientation Detection

This processing blockconsists in extracting the orientations of the main vessels. The retinal vessels merge from the DO vertically and horizontally where directions are between 45 ° and 135 °, respectively. Thus, the vessel structures are extracted in each directionamong α = {0 °; 45 °; 90 °; 135 °}. Since the vessel segments may have differentorientations, vessel in each direction α must be evaluated with different angles. Each angles φα for each direction is presented in equation (3).

$$\varphi 0° = \{0°, 15°, 30°, 45°, 135°, 150°, 165°\}; \quad (3)$$

$\varphi 45° = \{30°, 40°, 50°, 60°\};$

$\varphi 90° = \{45°, 60°, 75°, 90°, 105°, 120°, 135°\};$

$\varphi 135° = \{120°, 130°, 140°, 150°\};$

Since the author aims to enhance linear structures, a logical choice for a structuring element is a "line" with a variable length and a variable angle covering both the short and long vessels. The linear structure (M) is performed for all the lengths and the deviation angles φα, where the line lengths designated by l={5; 10; 15; 20;25} pixels. Thus, the first step in this processing block consists at applying Opening in order to extract the vessels having a shape similar to the elements of linear structure M. The application of this operator leadsv to conserve the vessels corresponding to the element structure (Mki)in each image OKi, where k is the length number and i is the angle number.The k * i = 80 images have the same resolution (M * N) of the image input RES. The second step, each image provided by Opening are compared to of the oneswith deviation φα, by fixing the length l. Then the vessel structure of each orientation {0°; 45°; 90°; 135°} is defined as the sum of the maxima obtained for each value of l. The approach of this step is illustrated by equation (4).

$$\alpha = \sum_{l=1}^{5} \max_{\varphi_\alpha} \boldsymbol{O}_l^{\varphi_\alpha} \qquad (4)$$

The third step leads to extract separately the horizontal and vertical coordinates of the OD position (px, py). For this purpose, two images IH and IV are created which correspond respectively to the horizontal axis and to the vertical axis.The first image IH contains the structure of the vertical vessels, which is determined by subtracting the orientations {0 °; 90}. Similarly, the second (IV) contains the structure of the vertical vessels, which is determined by adding the orientations {45 °; 90 °; 135 °}.

2.4 Vessel concentration («VC»)

This processing block consists at detecting the converging points and finding the concentration zone of the vessels network. Thus, the images IH and IV are subdivided successively d times along the vertical and horizontal direction, respectively. The maximum number of divisions dmax, is calculated such as formulated in equation (5).

$$d_{max} = \text{round}\left(\frac{\max(N,M)}{\mu}\right) \qquad (5)$$

Where μ=70 or 45 respectively for IH and IV images. At each subdivision d, the regions ri (i = 1; ...; d+1), disjoint vertical, are created on the image IH (each resolution (N, M / d + 1)).Then, a vertical average of each region moy_r_i is calculated based on the image IH and BW (binary mask image), as indicated in equation (6). Then, a vertical division image (dv) is created, whose pixel values are equal to the average value of the corresponding region.Finally, the images extracted at each subdivision are added together.Similarly, this is extended to image IV to create the horizontal division image (dh), as indicated in equation (7).

$$moy\_r_i = \frac{\sum_1^N \sum_i^{i*M/d} IH}{\sum_1^N \sum_i^{i*M/d} BW} \quad (6)$$

$$moy\_r_i = \frac{\sum_i^{i*N/d} \sum_1^M IV}{\sum_i^{i*N/d} \sum_1^M BW} \quad (7)$$

2.5 Post-processing

In this processing block, the OD region is determined based on the highest vessel concentration index.Therefore, a horizontal projection on the vertical division image (dv) is performed to identify the pxposition. Similarly, a vertical projection on the horizontal division image (dh) is performed to identify thepyone. The point selected as point [px; py], is marked in the retinal image as the location of OD.

In some retinal images, (px; py) may be slightly outside the OD region. To overcome these situations, the maximum point of vessel convergenceand the point of maximum intensity, designated respectively by (cx; cy) and (bx; by), are calculated within these regions. The final OD position is given by the average of the three points p(x,y),c(x,y) and b(x,y) (Fig.4).

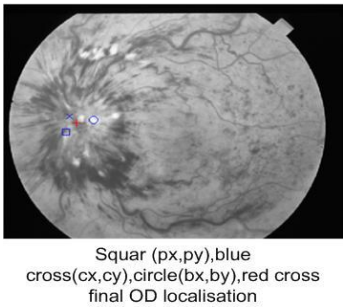


Fig. 4. Final OD localisation

3 Complexity analysis & parallelism principles

The objective of this section is to study the computing performance of the OD detection method proposed by Soares [7]. Therefore, the computational complexitiesare determined for each processing blocks and for each step of the vessel enhancement and the vessel orientation detection blocks, in terms of the input image resolution M × N. similarly, the execution times are provided by implementing the method in C ++ &OpenCV and run using STARE database images whose resolution is (605 * 700). The computational complexity in terms of approximate number of operations and execution time values are indicated in Table 1.

The post processing complexity is modeled in terms of w which corresponds to the OD diameter. Based on [1], the w value can be substituted by 1/7 M, which implies a

whole complexity equal to 245*M*N+3.5*M. The implementation leads to an execution time equal to 10.25s.
This approach is applied to image where the maximal resolution is 900 * 900 pixels. However, actual ophthalmologic devices provide fundus images an important higher resolution, such as the ones described in [12] where the resolution 4 to 7.4 times greater. Based on the whole computational complexity in terms of approximate number of operations, such rising on fundus image resolution implies a similar increase on the execution time. Consequently, the computational performance is a limiting factor to employ the OD method [7]. Therefore, parallelizing the OD detection processing is primordial in order to reduce execution time.

Table 1. OD detection treatments profiling.

		Complexity in terms of approximate number of operations	Execution time (seconds)
Preprocessing		5.NM	0,042
Vessels Enhancement	« blood vessel extraction»	3.NM	0,11
	« edge detection »	NM	0,06
	« LSC»	123. NM	4,14
Extract the 4 principal vessels	«Openning»	80. NM	2,57
	«Maximization & Addition»	NM	0.052
«VC»		31.NM	2.58
Postprocessing		NM+23.w	0,7
Total OD localization		245.NM+23w	10,25

Based on the method description in Section II, each step employs the result provided by the previous processing. Consequently, a parallelism on the processing step level is inadequate. All processing steps can be performed in (n × MN) instructions where n is an integer value n > 0. Each step corresponds to an iterative processing with a higher iteration number. Therefore, a parallelism strategy is able to be applied for each step separately. The SIMD principle is the adequate principle of parallelism where the implementation is to run on GPU architectures.

4 Processing optimization

Our first contribution aims to optimize the processing in order to reduce the execution time. The "LSC" processing consists at defining the H matrix of spatial derivatives for each pixel, as indicated in the equation (1) in section III.A. The Hessian matrix processing requires defining λ parameter by resolving the determinant of the matrix, for each pixel as indicated in equation (8).

$$\text{Det}\,(|\,H - \lambda A\,|\, = 0) \qquad (8)$$

Where A is the identity matrix. Thereafter, the formulations of λ1 and λ2 parameters are to be computed. The resolution of the matrix H and hence the formulation of λ1 and λ2 equations are done for each image pixel I whose number is M.N. However, those tasks are performed with the same size (2 * 2) of the H matrix. Therefore, λ1 and λ2 will have the same formulation whatever the pixel is, where their equations are indicated respectively in (9) and (10) [20, 21].

$$\lambda 1_{(i,j)} = \frac{1}{2}\left(-\sqrt{2 * Y * Z + Y^2 + 4 * T^2 + Z^2} + Z + Y\right)_{(i,j)} \qquad (9)$$

$$\lambda 2_{(i,j)} = \frac{1}{2}\left(\sqrt{2 * Y * Z + Y^2 + 4 * T^2 + Z^2} + Z + Y\right)_{(i,j)} \qquad (10)$$

Thereby, we proceed to determine the equations (8) and (9) only once in order to optimize the "LSC" processing time. The Figure 5 illustrates the flowcharts of the "LSC" step respectively before and after optimization, where the formulation steps of λ1 and λ2 modeled with a green background, are removed from the iterative loop. Equations λ1 and λ2 are provided as constant in the implementation. In this way, The "LSC" is performed in O(22MN) times instead of O(123xMN) times, where the input image size is (M*N).

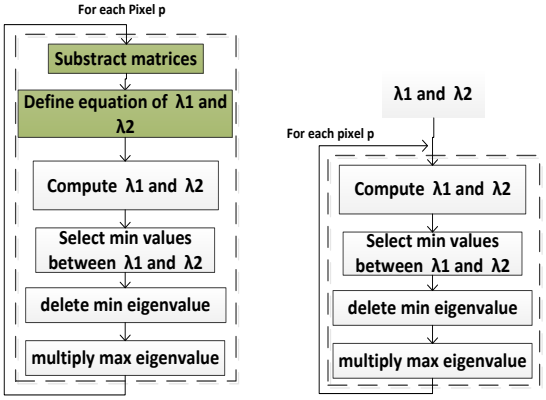


Fig. 5. Optimization of treatment «LSC» (a) Computingλ1 and λ2 inside loop (b) computing λ1 and λ2 outside loop.

The proposed Vesselness measure [33], Hessian multiscale features [34] and Feature Extraction [35] also based in the Hessian matrix eigenvalues. It can be observed that the optimization principle of "LSC" can be in applied order to reduce the execution time of these treatments.

For the vertical subdivision in the “VC” processing block, the column pixel sums Vv and Bv are performed respectively from the image IH and BW, as indicated in (Eq.6). Thereafter, the successive subdivision is applied. Those steps are repeated 10 times. However, V_v and B_v computing leads to the same results whatever the iteration

is. Similarly, the V_h and V_{bh} in the horizontal subdivision are performed 20 times to result identical values.

Therefore, we proceed to optimize the processing by computing once the Vh, Vv, Bv and Bvt vectors. Theirs processing are moved outside the loops, as modeled with a red background in Fig 6. In this way, the treatment «VC» is performed in O(MN+10.M+ 20.N) times.

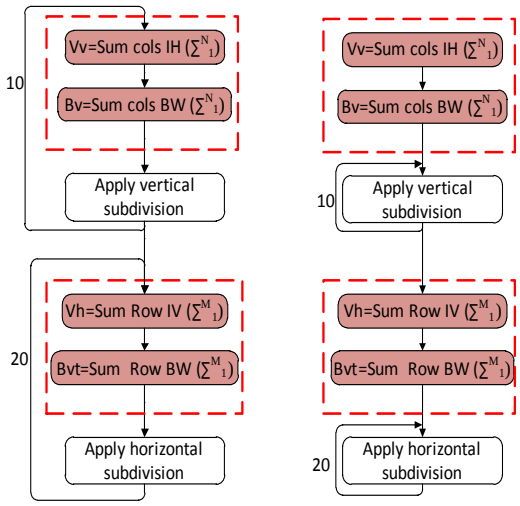


Fig. 6. Optimization of treatment «VC» (a) Computing Vv, Bv, Vh and Bvt inside loop (b) computing Vv, Bv, Vh and Bvt outside loop.

5 GPU implementation

In this section, we aim to parallelize the implementation of the pipeline image processing proposed by [7] on GPU architecture. In fact, the Opencv / GPU library proposes a predefined set of image processing functions that are run in GPU architecture, proposing a higher computational performance.

Therefore, our parallel implementation principle consists of implementing the steps using directly OpenCV/GPU library if they have corresponding functions. These processing steps are joined in the GPU architectures to avoid communication time between host and device. The intermediate data is directly integrated into the memory of the GPUs architecture. In the opposite case, the processing is implemented using CUDA kernel as described in the following section. The implementation of all steps is modeled in fig.7 where OpenCV/GPU functions and kernels are modeled respectively by yellow and red rectangles.

5.1 “LSC” processing kernel

“LSC”processing consists at computing the RES[i,j] in terms of the pixels X[i,j], Y[i,j], Z[i,j] and T[i,j] where Res in the output image, X, Y, Z and T in the intputs images. The “LSC” treatment can be processed independently for each pixel (N*M values), giving significant computational effort.

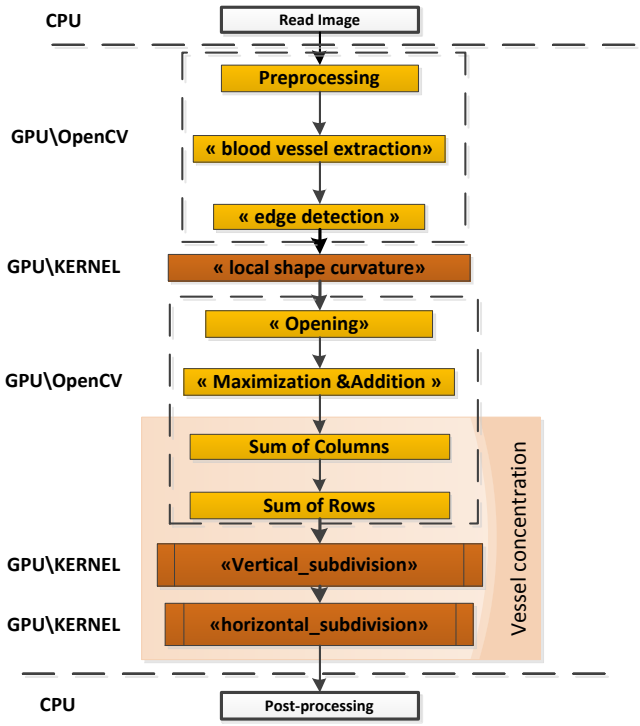


Fig. 7. Diagram block of the proposed GPU implementation.

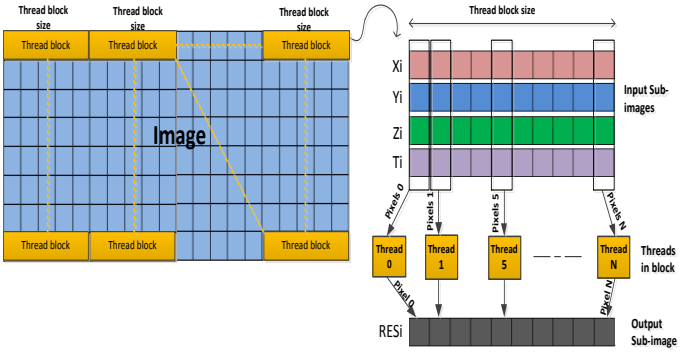
The result is also that all pixels can be processed simultaneously by independent computing threads.

we have previously uploaded the four images X,Y,Z and T, from the host memory to the device global memory and a CUDA kernel is launched to make the LSC treatment.

Firstly we designed an algorithm similar a ‘‘pixel by pixel’’ approach where each thread will do the computations concerning one pixel and add the resultat to Image RES. With this approach, The images are divided into several sub-images xi, yi, zi and ti and each is processed into a thread block, as indicated by the yellow cell in the image of Fig.8 (a). Each thread block (N threads) computes the LSC for a particular sub-image RESi of the result image, where i correspndante to the number of thread block is determined as indicated in equation (11).

$$NBblock = round\ (NBpixels\ /\ (NBSM * NBgpu)) \quad (11)$$

Thereafter, we proceed to parallelize the processing of the resulting image pixels. Therefore, each thread in the thread block provides single-pixel "LSC" processing of the resulting image. The pixels of the same indices of images X, Y, Z and T represent the parameters of single-thread Input, as indicated by cells in the input sub-images of Fig.8 (b).



(a) (b)

Fig. 8. Fragmentation of images: (a) Ordering on blocks of threads. (b) Ordering on threads

As a result, the complexity of the processing is reduced to the complexity of the algorithm executed in the kernel (LSC).

Algorithm 1 presents the kernel code executed by all threads, The KERNEL « LSC» will be called in the host by the instruction:

*__LSC__<<<NB_{block},N>>>(INPUT:*X,*Y,*Z,*T, cols, rows, OUTPUT:*RES);

Algorithm 1 : LSC KERNEL

```
_global_void LSC (IN :X,Y,Z,T ; OUT :RES)
tx ← thread x position within the block
λ1[tx]← Apply Eq_8(Y[tx],Z[tx],T[tx])
λ2[tx]← Apply Eq_9(Y[tx],Z[tx],T[tx])
min_values←MIN(λ1[tx], λ2[tx])
RES[tx] ←X[tx]-min(X[tx], min_values)
max_values←MAX(λ1[tx], λ2[tx])
RES[tx] ←RES[tx] * max_values
synchtreads() _Wait for all thread to finish their con-
tribution computations.
end KERNEL
```

5.2 «VC» processing kernel

The implementation of "VC" consists of creating subdivision images along the vertical and horizontal direction. The creations of the subdivision images are performed respectively according to the input images BW and IH or IV.

Based on the VC processing optimization in Section III, we note that the sums of the pixels of the columns of the images IH and BW **«Compute $\sum^{N}_{1}$(IH) & $\sum^{N}_{1}$ (BW) »**, are performed in parallel using the "cuda :: reduce ()" function of the OpenCV/CUDA library. The results are recorded respectively in the vectors Vv and Bv. The reduce() function can be used to compute horizontal and vertical sums of an image.Thus, the sum of the pixels of the lines of the IV and BW images **«Compute$\sum^{M}_{1}$(IV) & $\sum^{M}_{1}$(BW) »**, is performed using the same function.The results are recorded respectively in the vectors Vh and Bvt.

Therefore, the parallelism of the "VC" processing consists in using two consecutive kernels «Vertical_subdivision » and « horizontal_subdivision».

Based on section II.C (Eq (5)), the maximum division number dmax is computed based on the largest dimension of the input image. Thus the images are proportionally scaled in a way that the largest dimension is 900 pixels. Consequently, in the "vertical_subdivision" processing allows only 10 subdivisions. Similarly, in the "horizontal_subdivision" processing allows 20 subdivisions.

The implementation of the kernels on GPU involves the transfer of four vectors Vh, Vv, Bv and Bvt into the global memory of the GPU.

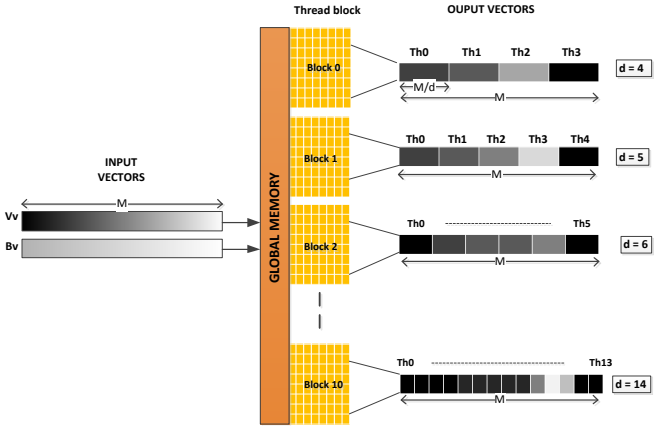


Fig. 9. Working on a GPU card, the first row shows Workflow of the « Vertical_subdivision ».

Vv and Bv are the input vectors for the «Vertical_subdivision » kernel. Firstly, the parallelism principle leads to perform the 10 subdivisions, which are applied along the vertical direction in parallel and each is processed into a thread block. The number of thread blocks is equal to the number of subdivision. Each thread block applies a single subdivision. Thereafter, at each vertical subdivision, we proceed to parallelize the processing of the creation of rp (p = 0; ...; d) disjunct regions. Each thread in the thread block is responsible for creating a single region by dividing the sum of vector region Vv by the sum of region Bv. The size of each region is calculated by dividing

the vector size Vv by the division number d corresponding to each thread block (d = blockIdx + 4), as shown in line 5 of the KERNEL «vertical_subdivision ». Consequently, parallel processing between threads decreases the times from O(10M) to O (M/4).

Algorithm 2 : Vertical_subdivision KERNEL

```
_global_void Vertical_subdivision (IN :Float* Vv,
Float* Bv, ,int cols; OUT :Float *dv)
tx ← thread x position
bx ← block x position
Idx←tx+ bx* block x dimensions
d← bx+4
TR ←cols/d;
for i = (Idx *TR) to ((Idx + 1)*TR) do
  If(i<cols)
  S1 ←S1+Vv[q]   %Somme of region in Vv vector
  S2 ←S2+ Bv[q] %Somme of region in Bv vector
  end if
end for
S3 ← S1 / S2
for i = (tx *TR) to ((tx + 1)*TR) do
   if (i<cols)
  dv[i + Idx*cols] ← S3
  end if
end for
 end KERNEL
```

In the second kernel «horiz_subdivision», Vh and Bvt are the input vectors for the subdivision application along the horizontal direction. Similarly, the same parallelism principle is performed for the 20 subdivisions. The main difference consists at modifying the line 6 by TR = Rows / d. The times of parallel processing between threads is decreased from O (20 N) to O (N/4).

6 Experimental Results

6.1 Experiment principles

As mentioned earlier, three contributions are used to reduce the execution time. After introducing the soft and hard environment employees described in session 6.2, we conducted three experiments to evaluate the execution time and the speed up of each contribution. The first experiment evaluates the impact of the algorithmic optimization on the OD detection execution time. The second experiment examines the GPU parallel implementation. Thus, we compare the performance of the Opencv/CPU functions and OpenCV/GPU ones. Hence, the impact of the kernel implementation is studied. Finally, we quantify the rising of the execution time of the whole implemen-

tation after the contributions. To insure a credible experimentation, the implementations are applied using 10 images of the STARE database, chosen randomly.

6.2 Hardware &software resources

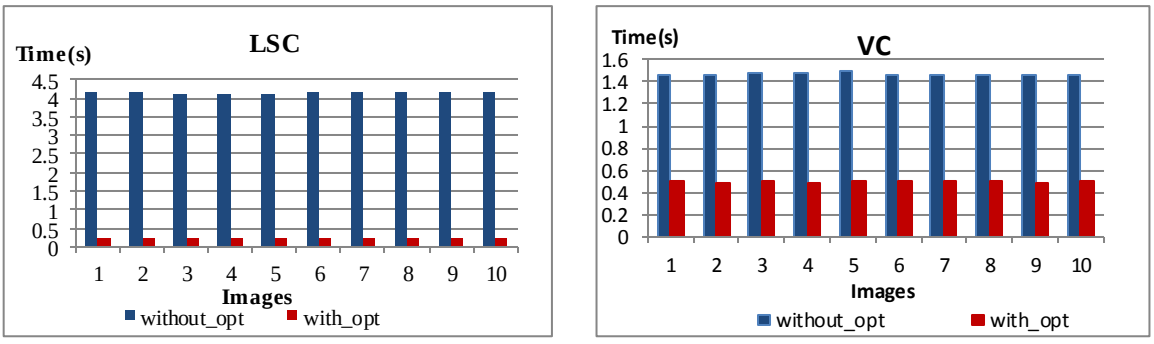
All implementations are tested on an I7 architecture having a processor frequency equal to 3.67 GHz with 8 GB of main memory and Windows 8.1 running. The parallel version is implemented using the CUDA v8.0 programming environment on NVIDIA Geforce GTX 980. This architecture belongs to the Maxwell family. It contains 16 streaming multiprocessors (SM). Each containing 128 processors (GPU) that operates at 1216 MHz.

OpenCV (Open Source Computer Vision) is an open source library originally developed by Intel, which provides functions for creating real time applications of computer vision and image processing. This library is written in C and C++ and can be run in environments such as Linux, Windows and Mac OS X. Initially, the implementation of the sequential OD detection algorithm was performed by combining the OpenCVversion 3.2 and C ++ programming languages.

In the parallel version, the operations are performed on the GPU using the appropriate OpenCV/GPU extension. OpenCV/GPU are open source libraries that provide an interface for video input, display and programming on GPU using a bunch of high-level implementations of various image processing and computer vision algorithms [22]. The processing time with OpenCV/cuda up to 18 times faster than native OpenCV function [23].

6.3 Algorithmic optimization evaluation

In this session, the algorithmic optimizations described in session V are evaluated. Each processing is coded before and after optimization, and run for all image set. The execution times are illustrated in Fig.10.(a) and Fig.10.(b) respectively for "LSC" and "VC". In this optimization phase an increased speedup, compared to the normal implementation phase, can be determined.



(a) (b)

Fig. 10. Execution time: (a) Without "LSC" optimized code Vs With"LSC" optimized code. (b) Without "VC" optimized code VsWith"VC" optimized code.

Then, the whole method is implemented before and after algorithmic optimization, where execution times are illustrated in Fig.11. This optimization allowed achieving averages speed up of 1.7.

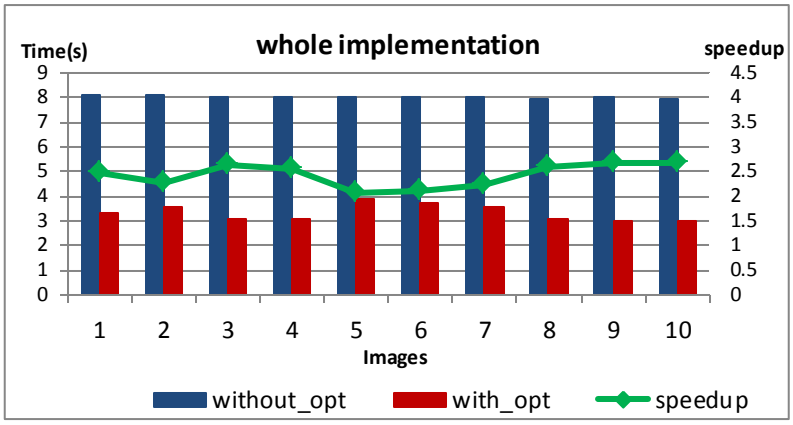


Fig. 11. Time needed to perform the whole OD detection algorithm without "LSC" & "VC"optimized code Vs. Time with "LSC" and "VC" optimized code.The speedup of the OD detection algorithm depending on the "LSC" and "VC" optimized code. The speedup is always given as relative to a CPU runtime.

6.4 GPU implementation evaluation

Evaluation of Implementing OpenCV functions on GPU architecture.

The OpenCV function set is implemented by default on the CPU and there after implemented in GPU-architecture. The execution times of the Opencv function set, on CPU and GPU are indicated in Fig.12.(a). Similarly, the whole execution time is shown in Fig.12.(b) where values prove that running OpenCV functions on GPU architecture leads to a speedup equal to 2.

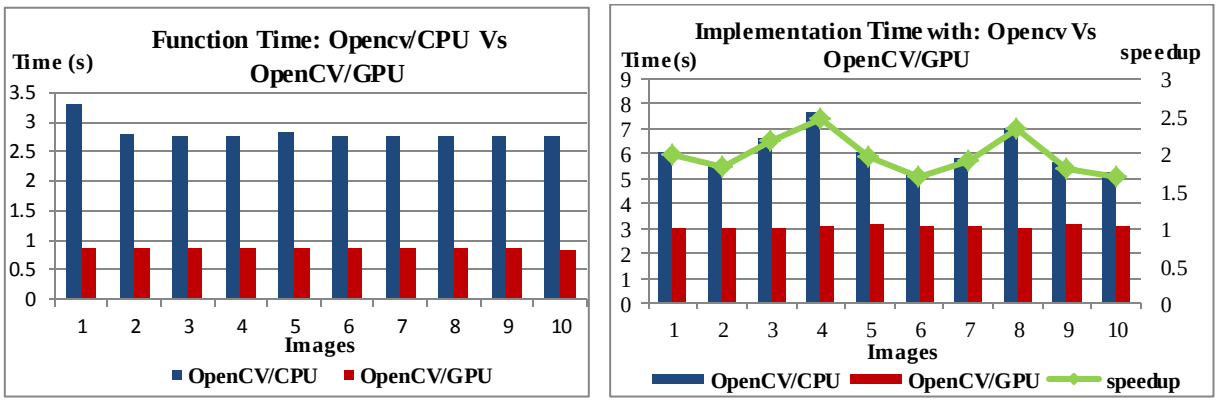


Fig. 12. (a)Time needed to perform all OpenCVfuction Vs. Time needed to perform all OpenCV GPU fuction. (b) Time needed to perform the whole optimized OD detection algorithm: With OpenCV Vs. With OpenCV-GPU. The speedup is given as relative to a GTX 980 runtime.

Evaluation of GPU kernels

To implement the "LSC" processing on GPU, we proceed to split the input image into 16 sub-images in order to distribute then to 16 blocks, with respect to the SMs number. Thus, each sub-image with (57 * 900) resolution is processed on a separate block. In fact, each SM is composed of Id.x *Id.y =128 GPUs. Thereby, each thread of the same block generates (1 * 900) pixels of the result image. For the distribution of images X, Y, Z and T between the threads, the pseudo code "*App_Hessien" will be called by the host with the following instructions:

```
*App_Hessien<<< 16, 128>>(INPUT:*X,*Y,*Z,*T, cols, rows,
OUTPUT:*RES);
```

The LSC processing is run on both CPU and GPU architectures where execution times are shown in Fig.13 (a). To implement the "VC" on GPU architecture, we proceed to perform consecutively tow kernels of subdivision image. For the kernel «verical subdivision », the maximum number of subdivisions performed on the vector Vh and Vv is less than the number of GPUs in a single SM of the GTX980 graphics card. Thereby, the processing of each subdivision is assigned to a single SM. In such a way, each thread consists of applying a single vertical subdivision. Similarly, in the second kernel «horiz_subdivision », the number of subdivisions performed horizontally does not exceed the number of GPUs in single SM. Thereby; each thread consists of applying a single horizontal subdivision.

The blood vessel concentration execution times in both CPU and GPU architectures are illustrated in Fig.13 (b). The implementation of both "LSC" and "VC"processing on GPU architecture allows to enhance considerably the execution times, where values is illustrated in Fig.13 (c). The execution time improvement leads to a speedup is equal to 2.5.

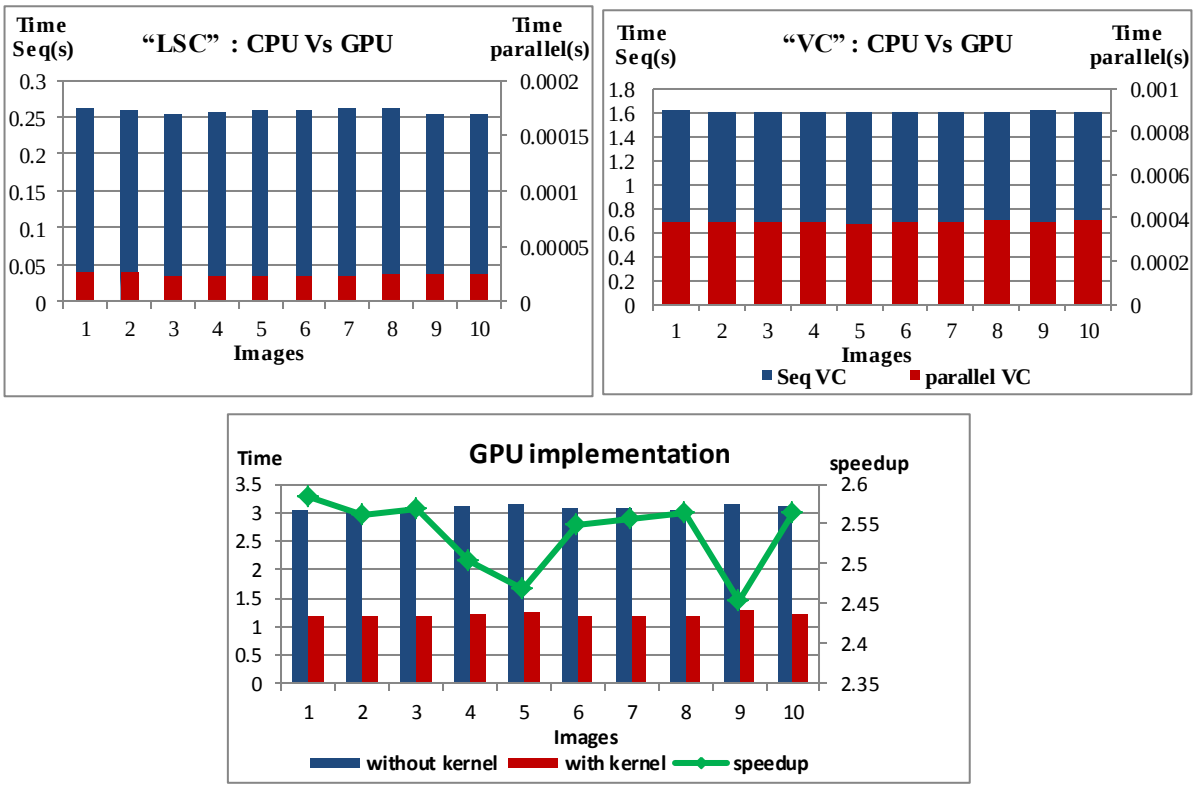


Fig. 13. Time needed to perform "LSC" sequential optimized code Vs "LSC" parallel optimized code. (b) Time needed to perform "VC" sequential optimized code Vs "VC" parallel optimized code.(c) Time needed to perform whole OD detection optimized algorithm with Opencv-cuda and with : the sequential "LSC" & "VC" code Vs. The parallel "LSC" and "VC" code. The speedup is given as relative to a GTX 980 runtime.

7 Conclusion

The paper objective consists at optimizing the method processing, proposed in [7], in order to enhance the computational performance.

First, we proceeded to optimize processing by shifting steps outside loops to reduce the time. Afterwards, all processing are implemented in GPU architectures.

Based on experimental results, the contributions insure enhancing computational performance where the speedups are respectively equal to 1.7 and 2.5. Then, all contributions provide a significant computational efficiency enhancement where the speedup is equal to 8.6, as indicated in Fig.14. Consequently, the average execution time is reduced from 10.2 to 1.1 seconds.

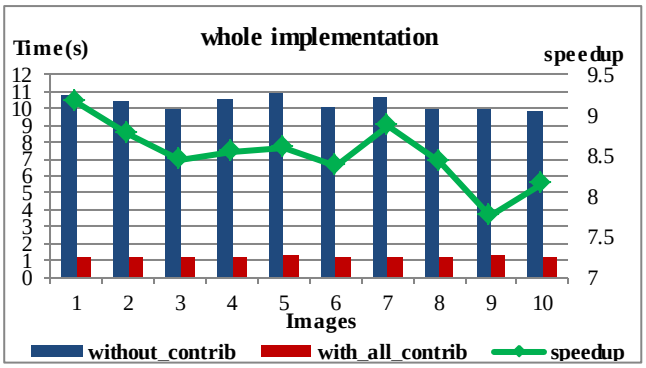


Fig. 14. whole implementation of OD detection: first version (c++/OpenCv version without optimization) vs. last version (parallel/Opencv-Cuda with Optimization).

The robustness of the technique proposed in [7] is guaranteed by evaluating the method in eight publicly-available datasets. Experiments revealed an OD localization accuracy of 99.15%, the proposed model reduces the average computation time 8.6 times when compared to the sequential method

References

1. S. BEN SAYADIA, Y. ELLOUMI, M.AKIL, M.H. BEDOUI, "Computational Efficiency of Optic Disk Detection on Fundus Image: A survey", SPIE Proceeding on Real-Time Image and Video Processing 2018, 15 - 19 April 2018, Orlando, Florida, United States, DOI:https://doi.org/10.1117/12.2304941
2. WorapanKusakunniran, Qiang Wu, PanraseeRitthipravat, Jian Zhang, HardExudates Segmentation based on Learned Initial Seeds and Iterative Graph Cut, Computer Methods and Programs in Biomedicine (2018), DOI: 10.1016/j.cmpb.2018.02.011
3. PavlePrentašić, Sven Lončarić, Detection of exudates in fundus photographs using deep neural networks and anatomical landmark detection fusion, Computer Methods and Programs in Biomedicine (2016), DOI:http://dx.doi.org/doi:10.1016/j.cmpb.2016.09.018.
4. askiratKaur , Deepti Mittal, A generalized method for the segmentation of exudates from pathological retinal fundus images, Biocybernetics and Biomedical Engineering, Volume 38, Issue 1, 2018, Pages 27-53, DOI:https://doi.org/10.1016/j.bbe.2017.10.003
5. Javeria Amin, Muhammad Sharif, MussaratYasmin, Hussam Ali, Steven Lawrence Fernandes, A Method for the Detection and Classification of Diabetic Retinopathy Using Structural Predictors of Bright Lesions, Journal of Computational Science. DOI:https://doi.org/10.1016/j.jocs.2017.01.002

6. SudeshnaSilKar, Santi P. Maity, Detection of neovascularization in retinal images using mutual information maximization, Computers and Electrical Engineering. DOI:https://doi.org/10.1016/j.compeleceng.2017.05.012
7. Soares I, Castelo-Branco M, Pinheiro AM. , Optic Disc Localization in Retinal Images based on Cumulative Sum Fields, IEEE J Biomed Health Inform. 2016 Mar.PP. 574-85. DOI: 10.1109/JBHI.2015.2392712.
8. Shuang Yu, Di Xiao, Yogesan Kanagasingam, Machine Learning Based Automatic Neovascularization Detection on Optic Disc Region. IEEE Journal of Biomedical and Health Informatics.2018. DOI:10.1109/JBHI.2017.2710201.
9. Soorya M.a, Ashish Issacb, Malay Kishore Dutt , "An automated and robust image processing algorithm for glaucomadiagnosis from fundus images using novel blood vessel tracking and bendpoint detection ", International Journal of Medical Informatics , November 2017. DOI: https://doi.org/10.1016/j.ijmedinf.2017.11.015
10. Fengshou Yin1, Jiang Liu1, Damon Wing Kee Wong1, Ngan Meng Tan1, Carol Cheung2, Mani Baskaran2,Tin Aung2 and Tien Yin Wong2 , "Automated Segmentation of Optic Disc and Optic Cup in Fundus Images for Glaucoma Diagnosis" , IEEE ,2012. DOI: 10.1109/CBMS.2012.6266344
11. KittipolWisaeng,Worawat Sa-ngiamvibool, Improved fuzzy C-means clustering in the process of exudates detection using mathematical morphology, April 2018, Volume 22, Issue 8, pp 2753–2764. DOI: https://doi.org/10.1007/s00500-017-2532-8.
12. Xavier Zanlonghi., "Un comparatif de rétinographes non mydriatiques, " Slate, 2005, http://www.ophtalmo.net/bv/Doc/2005-5918-XZ-RNM.pdfasp (20 Mars 2018).
13. Jun Cheng, Jiang Liu, Yanwu Xu, Fengshou Yin, Damon Wing Kee Wong, Ngan-Meng Tan, Dacheng Tao,"Superpixel Classification Based Optic Disc and Optic Cup Segmentation for Glaucoma Screening",IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 32, NO. 6, JUNE 2013.DOI: 10.1109/TMI.2013.2247770
14. Jyoti Prakash Medhi , Samarendra Dandapat : "An effective fovea detection and automatic assessment of diabetic maculopathy", Computers in Biology and Medicine, journal (Computers in Biology and Medicine 2015) 1-15.
15. Ravi Kamble, Manesh Kokare, Girish Deshmukh, Fawnizu Azmadi Hussin, Fabrice Mériaudeau :"Localization of Optic Disc and Fovea in Retinal Images using Intensity Based Line Scanning Analysis",journal (Computers in Biology and Medicine 2017)1-29.
16. R. GeethaRamani-2018, Macula Segmentation and Fovea Localization employing Image Processing and Heuristic based Clustering for Automated Retinal Screening.
17. Sudeshna Sil Kar, Santi P. Maity ;Retinal blood vessel extraction using tunable bandpass filter and fuzzy conditional entropy
18. M. Usman Akram, Shehzad Khalid, Anam Tariq, M. Younus Javed, Detection of neovascularization in retinal images using multivariatem-Mediods based classifier, Computerized Medical Imaging and Graphics .
19. Fully Automatized Parallel Segmentation of the Optic Disc in Retinal Fundus Images ,Daniel D«ıaz-Pernil et al.
20. [book, 20] biological and medical physics, biomedical engineering.
21. CURVATURE DETECTION AND SEGMENTATION OF RETINAL EXUDATES.
22. GpuCV: A GPU-Accelerated Framework for Image Processing and Computer Vision YannickAllusse
23. Point to point processing of digital images using parallel computing ,EricOlmedo
24. Superpixel Classification Based Optic Disc and Optic Cup Segmentation for Glaucoma Screening, Jun Cheng et al.

25. R.Pourreza.S ,MeysamTavakoli and Nasser Kehtarnavaz, "Computationally efficient optic nerve head detection in retinal fundus images, "BiomedicalSignal Processing and Control. 63–73(2014).
26. F. A. Hashim, N. M. Salem, and A. F. Seddik. "Optic Disc Boundary Detection from Digital Fundus Images. " Journal of Medical Imaging and Health Informatics, 50–56 (2015).
27. S.Giraddi, JagadeeshPujari and P.S.Hiremath., "Optic Disc DetectionUsing Geometric Properties and GVF snake. " Intelligent Systems and Information Management (2017).
28. DongboZhang and YuanyuanZhao. "Novel Accurate and Fast Optic Disc Detection in Retinal Images With Vessel Distribution and Directional Characteristics"; IEEE Journal of Biomedical and Health Informatics,333–342 (2016).
29. M. Foracchia, E. Grisan, A. Ruggeri, "Detection of optic disc in retinal images by means of a geometrical model of vessel structure", IEEE Trans. Med. Imaging. 1189–1195 (2004).
30. Li Xiong and HuiqiLi, "An approach to locate optic disc in retinal images with pathological changes", Computerized Medical Imaging and Graphics, 40–50 (2016).
31. A. Youssif, A.Z. Ghalwash, and A.R. Ghoneim, "Optic disc detection from normalized digital fundus images by means of a vessels' direction matched filter, " IEEE Trans.Med.Imaging.11–18 (2008).
32. Khai Sing Chin, Emanuele Trucco , Lailing Tan , Peter J. Wilson :Automatic fovea location in retinal images using anatomical priors and vessel density; journal (Pattern Recognition Letters2013)1-7.Tolias, Y., Panas, S., 1998. "A fuzzy vessel tracking algorithm for retinal based on fuzzy clustering", IEEE Trans. Med. Imaging 17, Iss.2, pp. 263-273.
33. Shahab Aslani , Haldun Sarnel "A new supervised retinal vessel segmentation method based on robust hybrid features", Biomedical Signal Processing and Control 30 (2016) 1–12.
34. Sumathi Thangaraj, Vivekanandan Periyasamy, Ravikanth Balaji, "Retinal vessel segmentation using neural network", IET Image Processing (2018), pp. 669-678.
35. Chengzhang Zhuc, Beiji Zoua, Rongchang Zhao, Jinkai Cui, Xuanchu Duan, Zailiang Chen, Yixiong Liang, "Retinal vessel segmentation in colour fundus images using Extreme Learning Machine", Computerized Medical Imaging and Graphics (2016), DOI: 10.1016/j.compmedimag.2016.05.004.