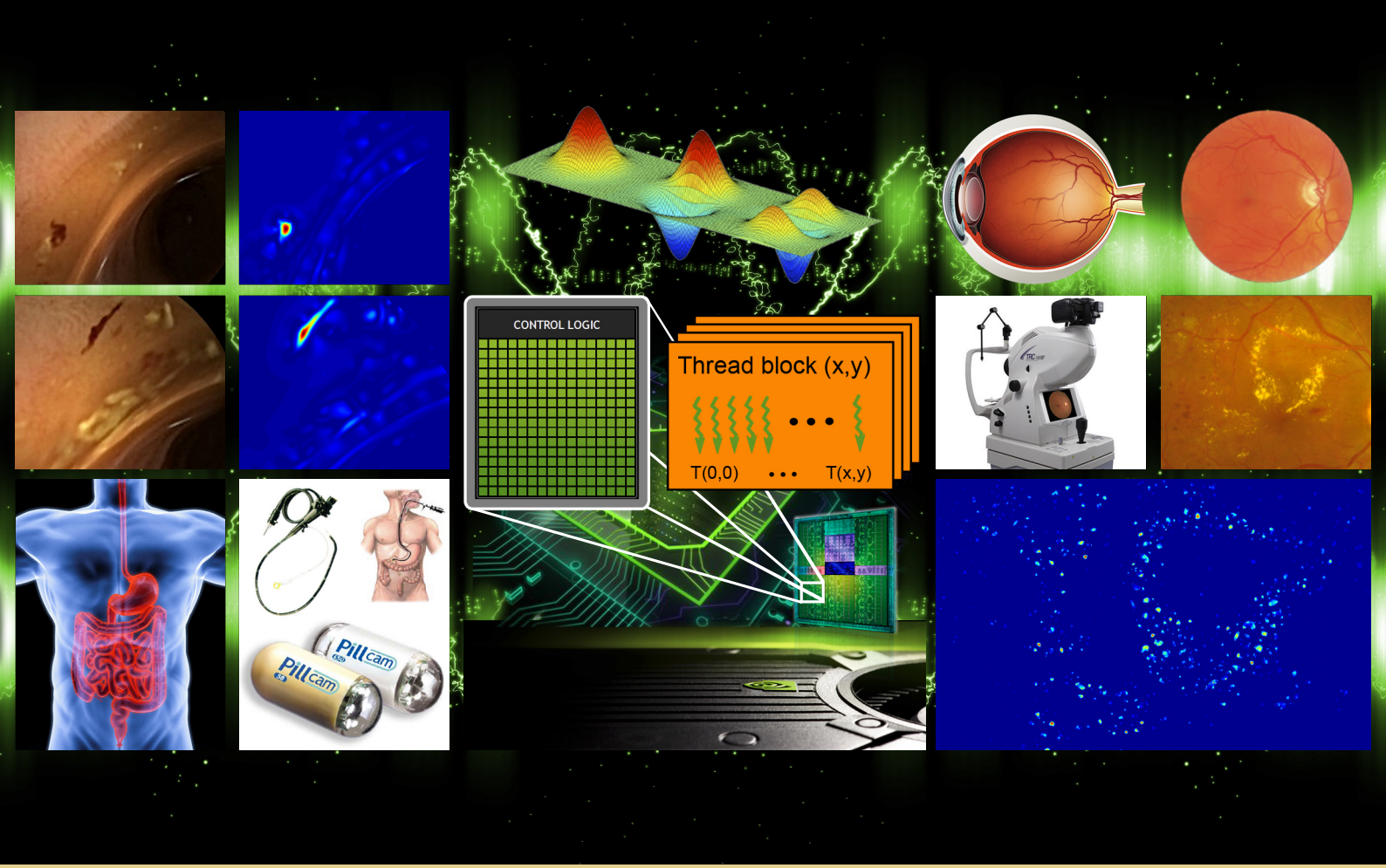


Carlos André Faria da Graça

Acceleration of computer-assisted image inspection algorithms in medical scenarios

Dissertação de Mestrado em
Engenharia Electrotécnica e de Computadores

Setembro de 2014

· U C ·

Universidade de Coimbra

Acceleration of computer-assisted image inspection algorithms in medical scenarios

Carlos André Faria Da Graça

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador:	Doutor Gabriel Falcão Paiva Fernandes
Co-Orientadora:	Doutora Isabel Maria Narra de Figueiredo

Júri

Presidente:	Doutor João Pedro de Almeida Barreto
Orientador:	Doutor Gabriel Falcão Paiva Fernandes
Vogais:	Doutor Jorge Nuno de Almeida e Sousa Almada Lobo

Setembro de 2014

Agradecimentos

Gostaria de começar por agradecer ao meu orientador, o Doutor Gabriel Falcão pelo seu acompanhamento constante do meu trabalho, pelas ideias propostas para melhorar o projecto e pelo tempo dispensado a responder a inúmeras questões. Agradeço também a minha co-orientadora Doutora Isabel Narra Figueiredo e ao Doutor Sunil Kumar pela disponibilidade para esclarecer qualquer tipo de dúvidas acerca do projecto.

Gostaria de dar uma nota de destaque também aos colegas de laboratório que se dispuseram para me auxiliar sempre que fosse necessário, a todos os meus amigos, pelo apoio proporcionado pelo facto de estarem perto, por terem manifestado interesse no meu trabalho, e estarem dispostos a ouvir-me explicá-lo repetidamente.

Agradeço também à minha mãe e ao meu pai, à minha irmã, bem como a toda a minha família, que sempre me apoiaram, em todo o meu percurso académico.

A todos vocês, um grande obrigado.

Abstract

Over the last years, image processing methods have been used with great success in the diagnosis of medical diseases. Good examples can be found in the processing of images regarding the gastrointestinal tract or Retinal Fundus (RF) images. A huge number of images is generated by new diagnosis methods, whose time consuming analysis needs to be carried out by a doctor. Recently, a new computer-aided diagnosis system for blood detection in Wireless Capsule Endoscopy (WCE) images has been proposed. The algorithm is composed of two phases: segmentation (for discarding uninformative regions in the image that can interfere with the blood detection); and the construction of an appropriate shape-based detector function, aiming the detection of blob and tubular shapes. This shape-based detector was also applied to High Definition (HD) RF images, in order to detect Exudates (EXs), which are yellow lipid deposits identified as bright yellow lesions. The presence of EXs is a good identifier for Diabetic Retinopathy (DR). This segmentation and shape-based detector still does not serve the purpose of performing the analysis within a small amount of time in any of the case studies presented. However, these methods can indeed be parallelized using Graphics Processing Units (GPUs). In this work, sequential and parallel image processing techniques are studied with emphasis on the filtering procedure in order to obtain a faster implementation. The filtering procedure has been identified as being responsible for up to 98% of the global processing time, in the shape-based detector functions (blood detection in WCE images). So, we implemented several versions of the filtering process and we concluded that separable time-domain approaches executing on GPU are faster for small filters and that frequency-domain GPU methods are more efficient for larger filters. By using parallel computing, a suitable single-GPU and an adaptative hybrid multi-GPU framework is proposed for speeding up the segmentation and shape-based detection procedures. It is shown that in the blood detection on WCE images the accelerated procedure running on a fast single-GPU version is on average 92 times faster than the original sequential Central Processing Unit (CPU) version, and is able of processing 119 frames-per-second (FPS). The proposed hybrid GPU-GPU system with Dual GPU NVidia GTX TITAN shows to be capable of processing 218 fps, which allows that the approximate total number of 56000 frames,

generated by a complete WCE exam (8 hours), can be computed in less than 5 minutes. In HD RF images only shape-based object detection is used, and the fastest single-GPU system can process 16 FPS with an average speedup of 179 times compared to sequential CPU version. In the proposed hybrid GPU-GPU system we can process 30 FPS with an average speedup 324 times faster than the original CPU version. With such high throughputs we are able to build real-time systems to automatically detect bright lesions in fundus images and blood in WCE images, which may help the medical practitioner improving the diagnosis procedure.

Keywords

Object shape recognition, Wireless Capsule Endoscopy, Exudates detection, Parallel image processing, Hybrid GPU-GPU Systems, CUDA

Resumo

Nos últimos anos, métodos de processamento de imagem têm sido utilizados com grande sucesso em diagnóstico médico. Encontramos bons exemplos disso, quer no processamento de imagens do trato gastrointestinal quer imagens da retina humana. Um grande número de imagens é gerado pelos novos métodos de diagnóstico, cuja análise deve ser realizada por um médico, tornando o processo bastante demorado. Recentemente, foi proposto um novo sistema de diagnóstico auxiliado por computador para deteção de sangue em imagens obtidas por *Wireless Capsule Endoscopy* (WCE). Este algoritmo é composto por duas fases distintas: a segmentação (para descartar regiões não informativas na imagem) e a construção de um detetor baseado na forma. Este detetor baseado na forma, também foi aplicado a imagens da retina, afim de detetar Exsudatos (EXs). EXs são depósitos lipídicos identificados como lesões amarelas brilhantes e a sua presença é considerada como um bom identificador para detetar a Retinopatia Diabética (DR). Os processos de segmentação e deteção ainda não servem o propósito de realizar uma análise rápida para qualquer um dos casos de estudo apresentados. No entanto, estes métodos podem ser paralelizados utilizando Unidades de Processamento Gráfico (GPUs). Neste trabalho, são estudadas técnicas de processamento de imagem sequenciais e paralelas com ênfase sobre o processo de filtragem a fim de obter uma implementação mais rápida. O processo de filtragem foi identificado como responsável por 98% do tempo global de processamento, nas funções do detetor baseada na forma. Neste sentido, implementamos várias versões do processo de filtragem, concluindo que o método separável no domínio do tempo executado em GPUs é o mais rápido para filtros pequenos e que os métodos no domínio da frequência na GPU são mais eficientes para filtros maiores. Tirando partido de computação paralela, um sistema single-GPU (NVidia GTX TITAN) e um sistema híbrido multi-GPU são propostos para acelerar os procedimentos de segmentação e deteção. Mostra-se que na deteção de sangue nas imagens WCE, o sistema mais rápido nas versões single-GPU é, em média, 92 vezes mais rápido do que a versão sequencial original Unidade de Processamento Central (CPU), sendo capaz de processar 119 imagens por segundo (FPS). O sistema híbrido GPU-GPU proposto com duas GPUs NVidia GTX TITAN revela-se capaz de processar 218 FPS, permitindo

que aproximadamente 56000 imagens, geradas por um exame WCE completo, possam ser processadas em menos de 5 minutos. Em imagens HD da retina humana, apenas a deteção de objetos com base em forma é aplicada, sendo que, o sistema mais rápido single-GPU consegue processar 16 FPS tornando-se, em média, 179 vezes mais rápido quando comparado com a versão sequencial CPU. No sistema híbrido GPU-GPU proposto, podemos processar 30 FPS sendo o sistema em média 324 vezes mais rápido do que a versão original na CPU. Com os resultados obtidos, somos capazes de construir sistemas de tempo real para detetar automaticamente lesões brilhantes da retina humana e zonas de sangramento em imagens WCE, o que pode ajudar o médico na melhoria do processo de diagnóstico.

Palavras Chave

Reconhecimento de objectos baseado na forma, Capsula Endoscópica, Deteção de Exsudatos, Processamento paralelo de imagem, Sistemas híbridos GPU-GPU, CUDA

Contents

List of Figures

List of Tables

List of Acronyms

CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
GCC	GNU Compiler Collection
GPU	Graphics Processing Unit
RAM	Random Access Memory
WCE	Wireless Capsule Endoscopy
CE	Capsule Endoscopy
FPS	frames-per-second
DR	Diabetic Retinopathy
RF	Retinal Fundus
EXs	Exudates
HD	High Definition
FFT	Fast Fourier Transform

1

Introduction

Contents

This work has been developed aiming the detection of tubular and blob shapes, having as motivation the acceleration of medical image processing using Graphics Processing Units (GPUs). It was applied to two case studies, namely: blood detection in Wireless Capsule Endoscopy (WCE) images and Exudates (EXs) detection in Retinal Fundus (RF) images.

1.1 Motivation

Segmentation and shape-based detection are two fundamental problems in computer vision which have been a major focus of research activities. Segmentation is the process of partitioning an image into distinct regions containing pixels with similar attributes. Some of the applications of image segmentation consist of locating tumors, aberrant Crypt Foci and other pathologies, or used in satellite images (roads, forests, etc.), face and finger print recognition, among many others. We are also interested in shape-based object detection, in particular objects that have blob/tubular shapes. Detection of blob and/or tubular structures in images is an important step in the analysis of large-scale scientific data, as for example, detection of blood regions in WCE images, bright lesions in RF images, nodule detection in thorax x-ray images, enhancement of vascular structures, to name a few. Recently, a new computer-aided diagnosis system for blood detection in WCE images has been proposed [1]. The algorithm is composed of two phases: segmentation (for discarding uninformative regions in the image that can interfere with the blood detection); and the construction of an appropriate shape-based detector function, aiming the detection of blob and tubular shapes and was also applied to High Definition (HD) RF images in order to detect EXs, which are yellow lipid deposits identified as bright yellow lesions [2].

The first case-study addressed consists of WCEs, which consists of a noninvasive endoscopic procedure that allows visualization of the entire gastrointestinal tract, and in particular the small intestine, without sedation or anesthesia, difficult to reach by conventional endoscopies. Capsule Endoscopy (CE) is useful for detecting small intestine bleeding/blood, polyps, inflammatory bowel disease (Crohn's disease), ulcers and tumors, among many others. As the name implies, CE makes use of a swallowable capsule (PillCam SB capsule by *Given Imaging Ltd.* in 2000 [3]) approved by the U.S. Food and Drug Administration in 2001 that contains a 576×576 resolution video camera, a light source, batteries, and a radio transmitter (please see Figure 1.1). In about 8 hours, the capsule travels through the entire gastrointestinal tract, capturing thousands of images. Data is stored on a recorder attached to the patient, so that physicians can review the images offline and analyze the potential source of diseases.

A major problem with this new technology is that it consumes a considerable amount

Figure 1.1: Left image: image of the capsule. Right image: view inside the capsule (M2A model features: 1-Optical, Dome 2-Lens Holder, 3-Lens, 4-Ilumination LEDs, 5-CMOS (Complementary Metal Oxide Semiconductor), 6-Battery, 7-Transmisor ASIC, 8-Antenna)

of time to analyze the approximately 56,000 images generated by the examination of a single patient. In addition, some abnormalities may be missed because of their size or distribution due to eyestrain. Therefore, it is very important to design a computerized real-time method for inspecting capsule endoscopic images and thus provide valuable help to the medical practitioner. In recent years, we have seen some new developments in the automatic inspection of CE images, improving the detection of blood, ulcers, polyps, tumors, among many others (see [4–13] for related work).

The main challenge for CE use is obscure digestive bleeding [6, 8–10, 12]. In fact, in many of these cases, the source of the bleeding is located in the small bowel. However, often these bleeding regions are not reached by traditional endoscopic techniques because of the reduced diameter of the small intestine. That is why blood detection through endoscopy by WCE is so important. Utilizing Ohta color channel $A_1 = (R+G+B)/3$ (where R, G and B denote the red, green and blue channel, respectively, of the input image) [14], they employed the analysis of eigenvalues of the image Hessian matrix and a multiscale image analysis approach for designing a function to discriminate between blood and normal frames. The experiments show that the algorithm is quite promising in distinguishing between both types of frames. However, a conventional Central Processing Unit (CPU) is not able to process a high number of images produced by WCE examination of a patient, within a small stipulated amount of time. Nonetheless, the computations of the algorithm can indeed be parallelized, and thus, process the large dataset of generated images faster. In the algorithm we identified two crucial steps: segmentation (for discarding non-informative regions in the image that can interfere with blood detection); and the construction of an appropriate blood detector function, which ended up responsible for consuming most of the global processing time.

In the second case-study analyzed, the analysis consists of Diabetic Retinopathy (DR), the leading cause of blindness among middle-aged population, a common complication of the retina usually associated with diabetes. The central area of the retina which is usually darker in digital fundus is called the macula (see Figure 1.2a). When the macula is affected in diabetic patients, it can lead to diabetic maculopathy or diabetic macular edema which is also considered as an advanced stage of DR. The presence of EXs, is one of the best indicators of the presence of DR, therefore this work also focus on the detection of EXs lesions. The severity of DR is determined by spatial distribution of EXss, especially in relation to the fovea (the center of macula).

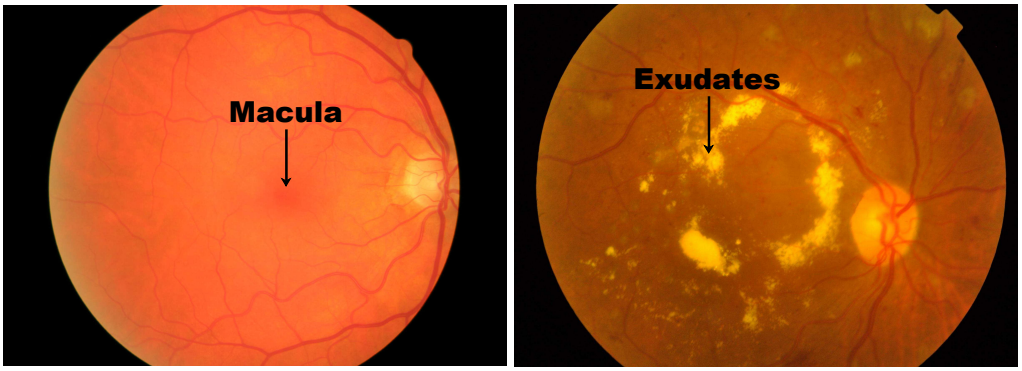


Figure 1.2: a) Normal retina fundus image. b) Exudates lesions visible in retina fundus image.

Recent years have witnessed some development on automatic inspection of RF images, improving the detection of exudates, glaucoma, vessel analysis, among others (see [2, 15–18] for related work). This shape-based detector was also applied on HD RF images, in order to detect EXs (please see Figure 1.2b).

This RF images are taken using a *Topcon TRC NW100 non-mydriatic* retinal camera, thus obtaining large HD images, which implies that the EXs detector computation represents a computationally time expensive task.

1.2 Objectives

The core of this thesis work is to exploit image processing techniques and the potential of parallel processing to implement versions of segmentation and shape-based detection applied to blood detection and EXs detection algorithms, using GPUs under the C++ code for parallel applications based on the Compute Unified Device Architecture (CUDA) framework [19]. The final goal consists of proving the feasibility of a WCE and EXs detection in medical imaging systems, achieving higher frame-rates than current systems, while exploiting the computational horsepower of multiple GPUs.

1.3 Main contributions

These algorithms (segmentation and shape-based detector) were originally developed and coded to run under MATLAB by the authors of [1] and they built all mathematical background who supports this applications. But, to support current medical imaging systems for blood detection in WCE images and EXs detection in RF images, massive computing power and higher frame-rates are required to achieve real-time systems. In this work, sequential and parallel image processing techniques are studied and compared running in different CPU and GPU platforms. Below, we highlight the main contributions of this work:

- *Accelerated Filtering Procedure*: We implemented and benchmark several versions of filtering procedure (conventional and separable approaches in time-domain versions and frequency-domain approach running C/C++ code on CPU and GPU). With the obtained results of this study, we select the fastest implementation in time-domain (Separable filtering method) and frequency-domain under CPU and GPU in order to perform more intensively benchmarks using small and large filters. This filtering benchmarks are performed under original ones and resized WCE images (in bigger images the size of objects (bleeding/blood regions) will be bigger implying the use of larger filters). After intensive testing, we conclude that separable time-domain approach executing on GPU are the fastest for small filters and frequency-domain approaches running on GPU are more efficient for larger filters. In CPU implementations, the filtering process is faster for small and larger filters using frequency-domain method with the FFTW3.3.3 library [20].

- *Hybrid GPU-GPU Computing Framework*: As a first approach, segmentation and shape-based object detection algorithms using parallel C/C++ code are developed to run on a single-GPU. Next, a suitable adaptative hybrid GPU-GPU framework is proposed for speeding up the segmentation and shape-based detection execution times even further. In hybrid GPU-GPU assemblies we exploit a balanced distribution of heterogeneous GPU resources, thereby deciding how many images are processed on each GPU. Our hybrid GPU-GPU algorithm automatically adapts to the available resources (CUDA devices), which means it is 100% portable across different machines who support CUDA, thats only need to run the training process once.

- *Real-Time System for Blood Detection*: By applying the segmentation and shape-based object detection on WCE images, experiments show that the accelerated single-GPU setup procedure is on average 92 times faster than the original one

executed on CPU and is capable of processing 119 frames per second. Our best hybrid GPU-GPU approach is on average 168 times faster than the original one (CPU version) and is able to process 218 frames per second. With the obtained speedup, our best hybrid GPU-GPU approach (Dual NVidia GTX TITAN) shows to be able to process a complete WCE exam (approximate total number of 56000 frames) in less than 5 minutes.

- *Real-Time System for Exudates Detection*: On HD RF images we only have to apply shape-based object detection, and the fastest single-GPU system can process 16 frames per second with an average speedup of 179 times. In hybrid GPU-GPU mode we are able to process 30 frames per second, with an average speedup of 324 when compared to the original CPU version.

The contribution of this work was recognized by the scientific community, having been accepted for publication two papers: *"Cooperative Use of Parallel Processing with Time or Frequency-domain Filtering for Shape Recognition"* in EUSIPCO 2014 conference [21] which is shown in Appendix A; and *"A GPU accelerated algorithm for blood detection in wireless capsule endoscopy images"* [22] was published at *Developments in Medical Image Processing and Computational Vision, Lecture Notes in Computational Vision and Biomechanics, Springer, 2014*, which is shown in Appendix B. We also submitted a journal article *"Hybrid GPU-GPU computing: accelerated kernels for segmentation and object detection with medical image processing applications"* to the *Journal of Real-time Image Processing*, which is shown in Appendix C.

1.4 Dissertation outline

This thesis is structured in six chapters. Following the introduction, chapter 2 focus on shape-based detection theory wich includes a suitable color space selection, the segmentation procedure and shape-based detector and visual results of applying the segmentation and detection procedures are shown. In chapter 3 the basic principles of GPU architecture are discussed, followed by an explanation of the parallel processing techniques. Chapter 4 features the experimental results and speedups obtained with the work developed. Finally, in Chapter 5, we address the conclusions of this work, while also providing a path for future work in this field.

2

Shape-Based Detection Theory

Contents

This chapter has the purpose of conveying the reader with the basic principles to segmentation and shape-based detection. Here, the segmentation function that is designed to remove uninformative regions such as bubbles, trash, dark regions and others, which can interfere with the detection of blood [1] in Wireless Capsule Endoscopy (WCE) images is presented. To detect blob-like and tubular-like structures a shape-based detector function is presented. With this detector it is possible to discriminate between blood and non-blood frames in WCE images and detect Exudates (EXs) in Retinal Fundus (RF) images.

Notation: Let Ω be an open subset of R^2, representing the image (or pixel) domain. For any scalar, smooth enough, function u defined on Ω, $\|u\|_{L^1(\Omega)}$ and $\|u\|_{L^\infty(\Omega)}$, respectively, denote the L^1 and L^∞ norms of u.

2.1 Blood Detection Color Space

The Ohta color space [14] is a linear transformation of the RGB color space. Its color channels are defined by $A_1 = (R+G+B)/3$, $A_2 = R-B$, and $A_3 = (2G-R-B)/2$.

They observe that channel A_1 has the tendency of localizing quite well the blood regions, as is demonstrated in Figure 2.2. The first row corresponds to the original WCE images with blood regions and the second row exhibits their respective A_1 channel images. They also want to note that, before computing the A_1 channel of the images, they applied an automatic illumination correction scheme [23] to the original images, to reduce the effect of illumination.

2.2 Exudates Detection Color Space

They observe that green channel (RGB color space) has the tendency of localizing quite well the exudates lesions on retina. By computing the Isotropic Undecimated Wavelet Transform (IUWT) of the green channel of the input image, they can see the exudates lesions with great emphasis (see the second row in Figure 2.4).

2.3 Segmentation Function

Uninformative regions such as bubbles, trash, dark regions and so on are very common in WCE images, which can interfere with the blood detection (see [4] for more information). In CIE Lab color space they observe that the second component (which they call henceforth a-channel) has the tendency of separating these regions from the infor-

mative ones. They first decompose the a-channel into geometric and texture parts using the model described in [24] for better removal of the uninformative regions, and perform the two phase segmentation. With a reformulation of the Chan and Vese variational model [24, 25], over the geometric part of the a-channel.

The segmentation method is described as follows: They first compute the constants c_1 and c_2 (representing the averages of I in a two-region image partition). Solving the following minimization problem

$$\min_{u,v}\left\{TV_g(u)+\frac{1}{2\theta}\|u-v\|^2_{L^2(\Omega)}+\int_\Omega\Big(\lambda\, r(I,c_1,c_2)\,v+\alpha\,\nu(v)\Big)\,dxdy\right\} \tag{2.1}$$

where $TV_g(u) := \int_\Omega g(x,y)|\nabla u|\,dxdy$ is the total variation norm of the function u, weighted by a positive function g; $r(I,c_1,c_2)(x,y) := \big(c_1-I(x,y)\big)^2-\big(c_2-I(x,y)\big)^2$ is the fitting term, $\theta>0$ is a fixed small parameter, $\lambda>0$ is a constant parameter weighting the fitting term, and $\alpha\,\nu(v)$ is a term resulting from a reformulation of the model as a convex unconstrained minimization problem (see [24, Theorem 3]). Here, u represents the two-phase segmentation and v is an auxiliary unknown. The segmentation curve, which divides the image into two disjoint parts, is a level set of u, $\{(x,y)\in\Omega:\, u(x,y)=\mu\}$, where in general $\mu=0.5$ (but μ can be any number between 0 and 1, without changing the segmentation result, because u is very close to a binary function).

The above minimization problem is solved by minimizing u and v separately, and iterated until convergence. In short they consider the following two steps:
1. v being fixed, they look for u that solves

$$\min_u\left\{TV_g(u)+\frac{1}{2\theta}\|u-v\|^2_{L^2(\Omega)}\right\}. \tag{2.2}$$

2. u being fixed, they look for v that solves

$$\min_v\left\{\frac{1}{2\theta}\|u-v\|^2_{L^2(\Omega)}+\int_\Omega\Big(\lambda\, r(I,c_1,c_2)\,v+\alpha\,\nu(v)\Big)\,dxdy\right\}. \tag{2.3}$$

It is shown that the solution of (2.2) is ([24, Proposition 3])

$$u=v-\theta\mathrm{div}p, \tag{2.4}$$

where div represents the divergent operator, and $p=(p_1,p_2)$ solves

$$g\nabla(\theta\mathrm{div}p-v)-|\nabla(\theta\mathrm{div}p-v)|p=0 \tag{2.5}$$

The problem for p can be solved using the following fixed point method

$$p^0=0,\ p^{n+1}=\frac{p^n+\delta t\nabla(\mathrm{div}p^n-v/\theta)}{1+\frac{\delta t}{g}|\nabla(\mathrm{div}p^n-v/\theta)|}. \tag{2.6}$$

Again from [24, Proposition 4], they have

$$v=\min\{\max\{u-\theta\lambda r(I,c_1,c_2),0\},1\}. \tag{2.7}$$

2.3.1 Segmentation Results

Figure 2.1 shows of the applied segmentation function. As expected, all uninformative regions such as bubbles, trash and dark regions, which can interfere with the detection of blood has been removed by this segmentation method.

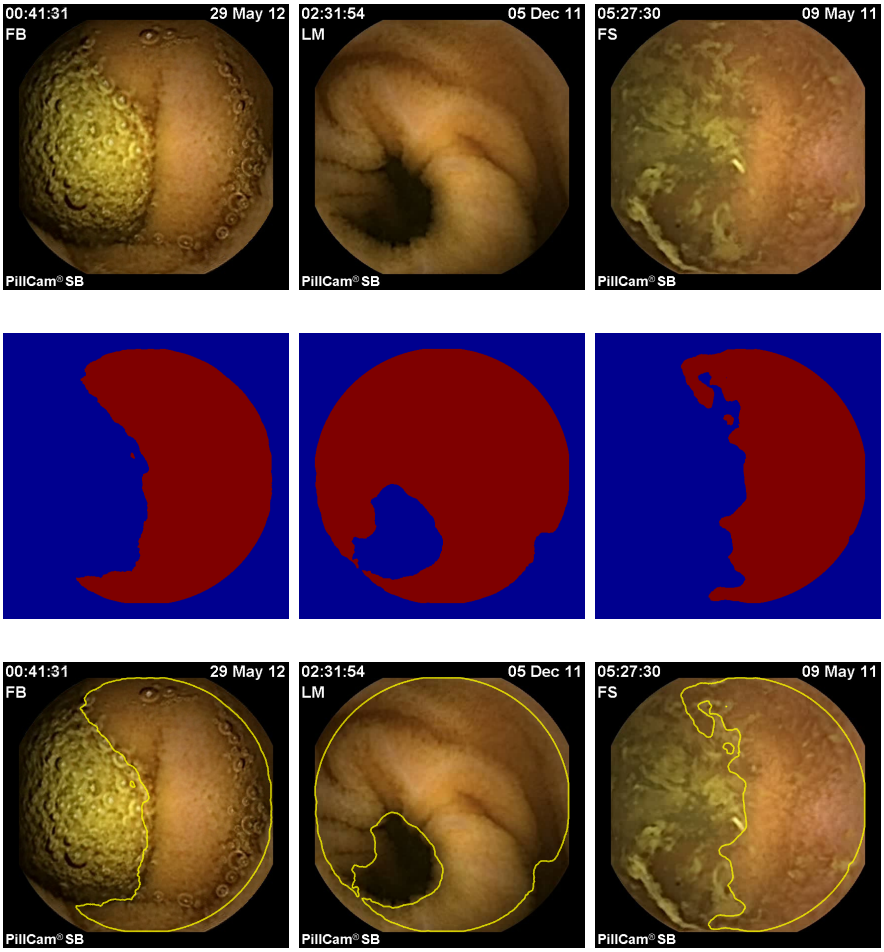


Figure 2.1: First row: Original WCE image where we can find uninformative regions like: bubbles on left image: dark regions on center image; trash in right image. Second row: Segmentation mask. Third row: Original WCE image with segmentation curve superimposed.

In these experiments the values chosen for the parameters involved in the definition of (2.1), are those used in [24], being g the following edge indicator function $g(\nabla u) = \frac{1}{1+\beta\|\nabla u\|^2}$ and $\beta = 10^{-3}$ [1].

2.4 Shape-Based Detector Function

Through the analysis of eigenvalues of the image Hessian matrix and multiscale image analysis approach. Based on the eigenvalues, both blob-like and tubular-like structures can be detected. For a scalar image $I : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}$, they define the Hessian matrix of one point (x,y), and at a scale s, by

$$H_s(x,y) = \begin{pmatrix} I^s_{xx} & I^s_{xy} \\ I^s_{xy} & I^s_{yy} \end{pmatrix}, \tag{2.8}$$

where I^s_{xx}, I^s_{xy} and I^s_{yy} are the second-order partial derivatives of I and the scale s is involved in the calculation of these derivatives. The Hessian matrix describes the second order local image intensity variations around the selected point. Suppose $\lambda_{s,1}$ and $\lambda_{s,2}$ are two eigenvalues of the Hessian matrix H_s. Further, suppose that $|\lambda_{s,1}| \leq |\lambda_{s,2}|$. Setting $F_s = \lambda^2_{s,1} + \lambda^2_{s,2}$, they define

$$F(x,y) = \max_{s_{min} \leq s \leq s_{max}} F_s(x,y), \tag{2.9}$$

where s_{min} and s_{max} are the minimum and maximum scales at which the blood regions are expected to be found. Remarking that, they can be chosen so that they cover the whole range of blood regions.

Setting now

$$f_1 = \exp\left(-\beta F_s^2\right) \quad \text{and} \quad f_2 = \left(1 - \exp\left(-\alpha \left(\frac{\lambda_{s,1}}{\lambda_{s,2}}\right)^2\right)\right), \tag{2.10}$$

and motivated from [26], they define the blob (B_s) and tubular (T_s) detectors (at each point of the domain)

$$B_s = \begin{cases} 0, & \text{if} \quad \lambda_{s,1}\lambda_{s,2} < 0 \quad \text{or} \quad |\lambda_{s,2} - \lambda_{s,1}| > \delta \\ (1-f_1)f_2, & \text{otherwise,} \end{cases} \tag{2.11}$$

and

$$T_s = \begin{cases} 0, & \text{if} \quad \lambda_{s,2} > 0, \\ (1-f_1)(1-f_2), & \text{otherwise.} \end{cases} \tag{2.12}$$

Here α and β are the parameters which control the sensitivity of the functions and δ is an user chosen threshold. Then, they compute the maximum for each scale

$$B(x,y) = \max_{s_{min} \leq s \leq s_{max}} B_s(x,y) \tag{2.13}$$

and

$$T(x,y) = \max_{s_{min} \leq s \leq s_{max}} T_s(x,y), \tag{2.14}$$

2.5 Blood Detector Configuration

In blood detection computations, they take $s = 8, 10, 12, 14$. The results of the functions F and the sum $B+T$, for blood and non-blood WCE images are displayed in Figures 2.2 and 2.3, respectively. Choosing $\alpha = 1000$ and $\beta = 1$ which control the sensitivity of the detector.

2.6 Exudates Detector Configuration

EXs detection algorithm is designed to discriminate between normal and non-normal frames in RF images. The EXs detector is very similar to detector blood. Here, the shape-base detector described in Section 2.4 is applied making minor configuration changes. Here, they only use the detection of blob-like structures (equation 2.11), using different scale values $s = [2\ 5\ 8\ 11\ 14]$ and choosing different values for $\alpha = 1$ and $\beta = 1$ which control the sensitivity of the detector. Then, they compute the maximum for each scale (equation 2.13).

The results of functions F and B, for normal and abnormal RF images are displayed in Figure 2.4, where we can identify EXs as sets of small bright dots.

2.7 Blood Detector Outline

Here we describe briefly the main steps of automated method for blood detection in WCE images proposed in [1]. Note that in the method both segmentation and shape-based object detection approaches are used appropriately. For each WCE image the method consists of the following main steps:

1. Firstly, they remove additional details (such as patient name, date and time) from the original image. For this purpose, they clip around the circular view of the original image.

2. They then consider the Ohta color channel $(R+G+B)/3$ for the input image.

3. They next apply the segmentation method (as described in Section 2.3) for removing uninformative regions (such as bubbles, trash, liquid, and so on) over the geometric part of the second component of the CIE Lab color space.

4. Finally, they compute the function $B+T$ (using the approach described in 2.4, where 8,10,12,14 are the values of the scale parameter s).

2.8 Exudates Detector Outline

The main steps of automated method proposed in [27] for bright lesions detection in fundus images are briefly described here. Note that in the method only shape-based object detection approach is used in an appropriate way. For each fundus image the method consists of the following main steps:

1. Firstly, they compute the Isotropic Undecimated Wavelet Transform (IUWT) of the green channel of the input image, and consider $W_2 + W_3 + W_4$, where W_j is the wavelet level at iteration j.

2. They then extract 20% highest pixels of $W_2 + W_3 + W_4$ and take the (pixelwise) product of the thresholded image with $W_2 + W_3 + W_4$ and remove the pixels corresponding to the optic disc.

3. Finally, from the resulting image, they compute the function $B + T$ (using the approach described in 2.4, where 2,5,8,11,14 are the values of the scale parameter s).

2.9 Blood Detection Results

The results of shape-based object detector function $B+T$, for some blood and non-blood images are displayed in Figures 2.2 and 2.3. For classification results of the blood detection method we refer the reader to [1, 22].

2.9.1 Blood Images

As shown in Figure 2.2, the blood detector works with good sensitivity. All blood regions are detected with acceptable accuracy.

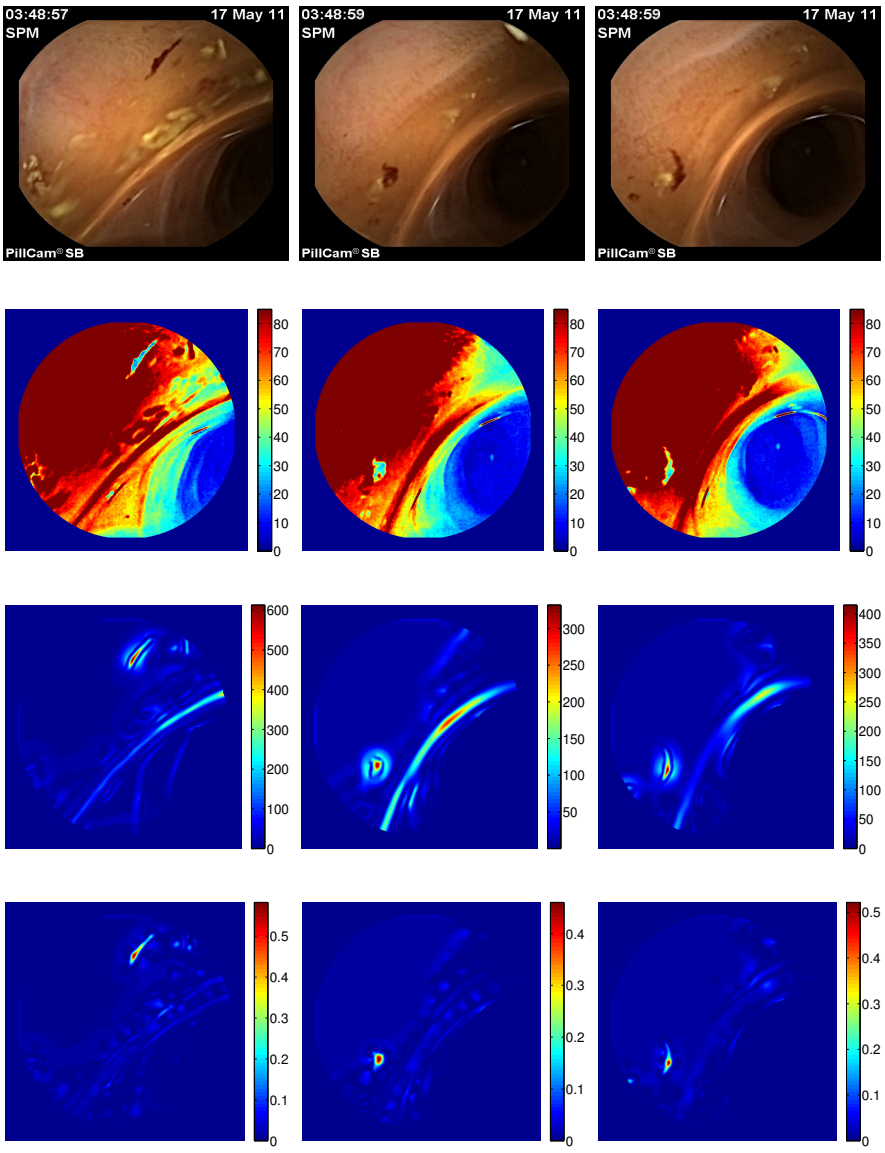


Figure 2.2: First row: Original WCE image with blood regions. Second row: A_1 color channel. Third row: Function F. Fourth row: Function $B+T$.

2.9.2 Non Blood Images

As depicted in Figure 2.3, we can see the blood detector results when applied to non blood WCE images. All images are classified like non-blood images (please see classification details in [1, 22]).

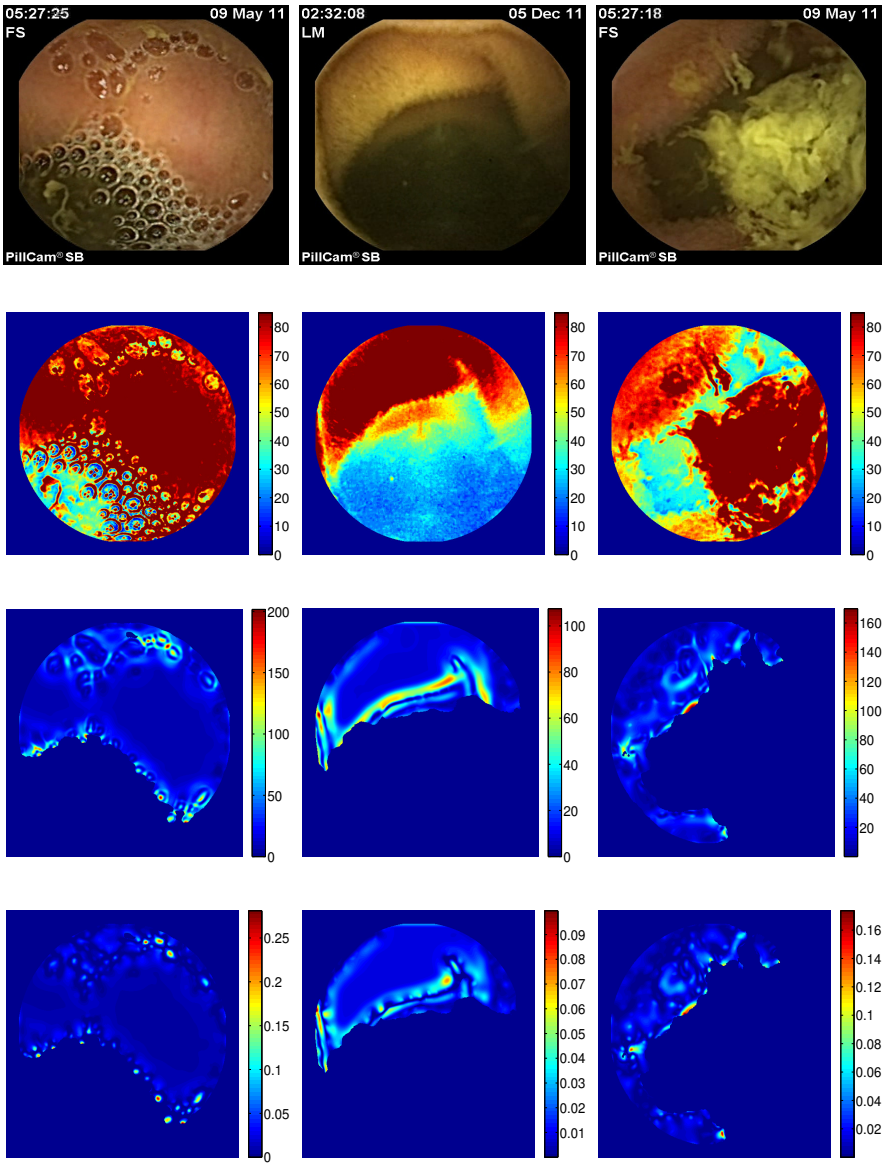


Figure 2.3: First row: Original WCE image without blood region, and where we can find uninformative regions like: bubbles on left image: dark regions on center image; trash impurities from the intestinal walls in right image. Second row: A_1 color channel. Third row: Function F. Fourth row: Function $B+T$.

2.10 Exudates Detection Results

The results of Wavelet candidates and function B, for normal and abnormal RF images are shown in Figure 2.4, where we can identify exudates as sets of small bright dots.

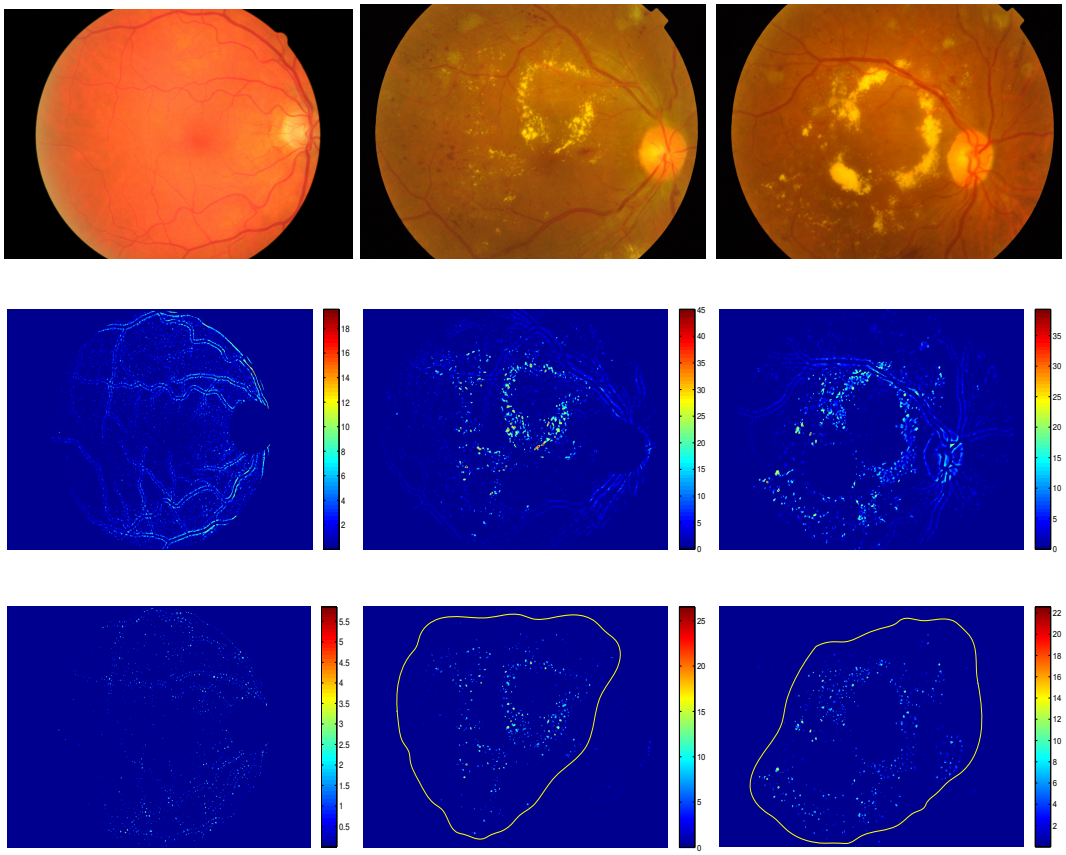


Figure 2.4: First row: Original retinal fundus image. Second row: Wavelet candidates (see step 2 in exudates detection outline 2.8). Third row: Function B. The result of the exudates detection when applied to three retinal fundus images are shown (normal image on first column; abnormal images on second and third columns). The exudates detector results, highlights the exudates as sets of small bright dots as shown in third row.

3

Parallelization of the algorithm

Contents

In this chapter we start by describing general facts about the apparatus specifications. In particular, we detail the Graphics Processing Units (GPUs) adopted and the underlying architectures. Finally, we address the parallelization of the algorithms proposed, namely by detailing the segmentation and shape-based detector parallelization procedures for the current medical datasets on the GPU.

The pipeline of the algorithm, described in Chapters 2.3 and 2.4, has been first implemented on a Central Processing Unit (CPU) Intel Core i7 950 @ 3.07GHz, with 12GB of Random Access Memory (RAM) clocked at 1600MHz, running a GNU/Linux kernel 3.8.0-31-generic. The C/C++ code was compiled using GNU Compiler Collection (GCC)-4.6.3. In order to process more frames-per-second (FPS), the segmentation and shape-based detection have been parallelized, for execute in these GPUs: NVidia Tesla C1060; NVidia Tesla C2050; NVidia GTX 680; NVidia GTX TITAN, compiled using NVIDIA Compute Unified Device Architecture (CUDA) driver 5.5 [19].

3.1 General overview of the GPU architecture

Generally the host system consists of a CPU which orchestrates the entire processing, sending data and launching parallel kernels on the GPU device. At the end of processing, the host collects the data calculated from the device and terminates execution. The parallelization of segmentation and blood detection procedures is carried out using the CUDA parallel programming model, by exploiting the massive use of thread- and data-parallelism on the GPU. CUDA allows the programmer to write in a transparent way, scalable parallel C code [19] on GPUs.

As shown in Figure 3.1, usually each thread processes one or more pixels (this depends on the algorithm; e.g., in 2D Separable Filtering using CUDA [28], each thread processes more than one pixel) and thus multiple elements can be processed at the same time. This represents a potential reduction in overall processing time of the proposed algorithm. When the host launches a parallel kernel, the GPU device executes a grid of thread blocks, where each block has a predefined number of threads executing the same code segment. All the threads execute synchronously and are time-sliced among the stream processors of each multiprocessor, as they run organized in groups of 32 threads (a warp).

A simplified overview of the GPU architecture is presented in Figure 3.2. As shown, several multiprocessors contain a certain number of stream processors (the number of stream processors and multiprocessors varies with the model and architecture of the GPU). In the present case, the GPU NVidia GTX TITAN contains 14 multiprocessors with each multiprocessor containing 192 stream processors, performing a total of 2688 CUDA cores, which allows to expect the faster execution performances of the algorithm.

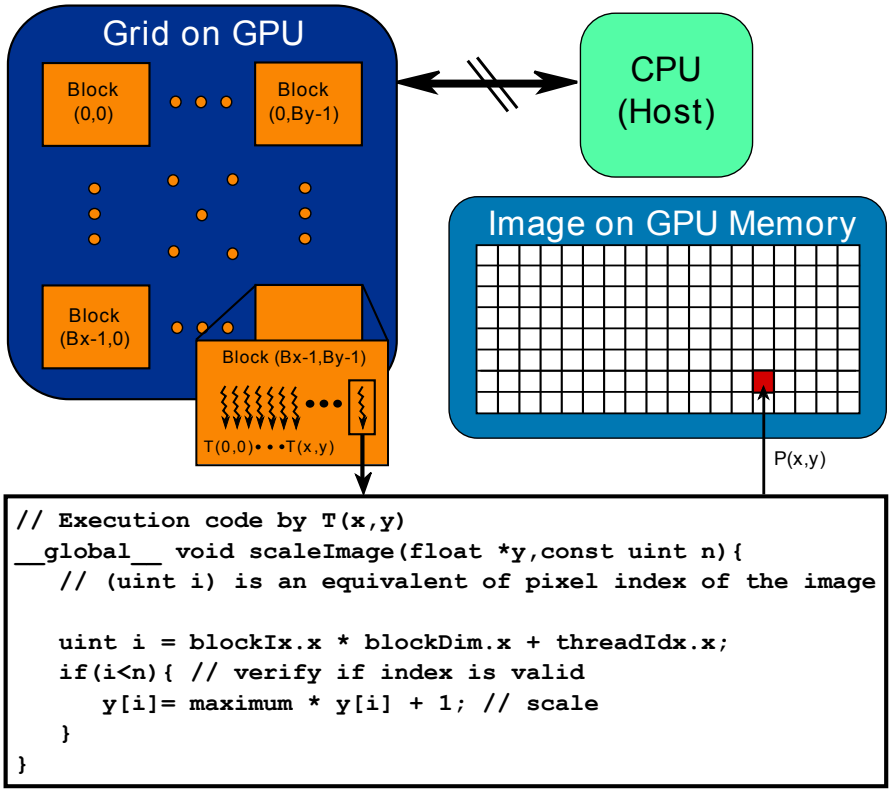


Figure 3.1: Illustration of the structure of a grid and thread blocks and how the same segment of code is executed by multiple threads. Each thread computes the result for one pixel.

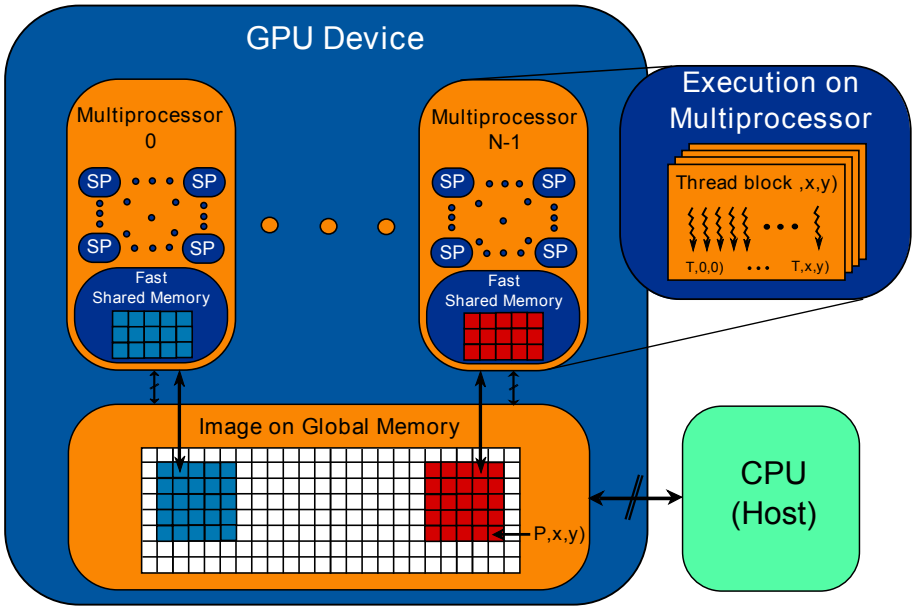


Figure 3.2: Simplified GPU arquitecture. An example of how thread blocks are processed on GPU multiprocessors. A multiprocessor can execute more then one thread block concurrently.

Before processing starts on the GPU, data is uploaded to device memory. This process is typically slow and consists in transferring the information from the host CPU memory to the GPU global memory (device). In opposition, at the end of processing, results must be transferred from the device memory to the host CPU memory.

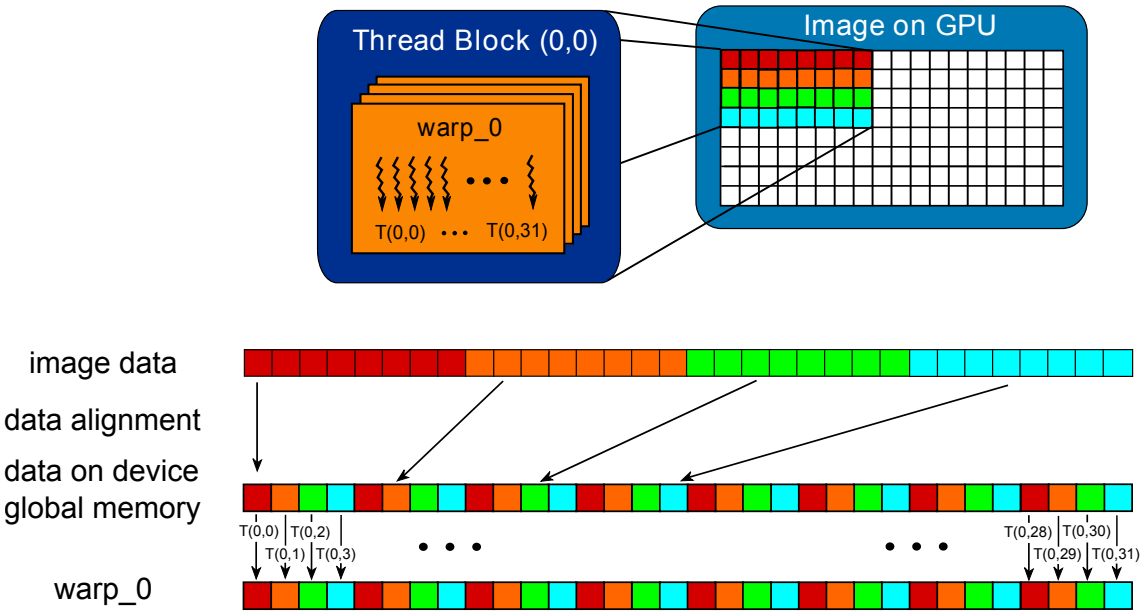


Figure 3.3: Coalesced memory accesses illustrating a warp of 32 threads reading/writing the respective 32 data elements on a single clock cycle.

In the GPU, there are several memory types and they have different impacts in throughput performance. We highlight two of them:

- Global memory accesses are time consuming operations with high latency and may represent a bottleneck in the desired system's performance. Thus, coalesced accesses should be employed whenever possible. They imply data in global memory to be contiguously aligned, so that all 32 threads within a warp can access the respective 32 data elements concurrently on the same clock cycle, with thread T(x,y) accessing pixel P(x,y), as depicted in Figure 3.3.

- Also, modern GPUs have small and fast blocks of memory tightly coupled to the cores, which is shared by all threads within the same blocks on multiprocessors. We can have several threads processing the same local data to optimize memory bandwidth (typically shared memory is faster than global memory when we need to share information among several threads), but shared memory is small in size. To optimize its use and performance, it is essential to consider these size limitations. Whenever large amounts of data have to be processed, data has to be partitioned in smaller blocks in order not to exceed the limits of shared memory. This procedure

can also represent penalties, since the amount of data exchanges with global memory in this case increases. Therefore, in the current work we use shared memory for calculating some procedures and only global memory to perform the remaining functionalities, globally achieving an efficient memory usage as reported later in chapter 4.

3.1.1 Simple Tips to Efficient Parallelization

The main goal of parallel computing is to minimize the execution time of a process, for which we have to use hardware resources efficiently. Thus, when designing a CUDA kernel, there are some techniques that we should be considered, since they optimize significantly the throughput performance of the program:

- *Asynchronous Memory Transfers*: Normally, one of the main performance bottlenecks in CUDA applications consists of data transfers between host (CPU) and device (GPU). In order to optimize data transfers and kernel executions it is possible to exploit CUDA streams and asynchronous memory transfers [19]. A stream is a sequence of commands that execute in order, and different streams can run concurrently. So, as an example we can create three different streams (for devices with two copy engines and one kernel engine) for each device (GPU) with different functionalities, which are: upload data to device; kernels execution; download data from device. By using this technique we can overlap memory transfers and kernel execution, thus increasing the overall throughput of the program.

- *Coalesced Memory Accesses*: When the kernel runs, and threads inside a warp need to perform a memory load, the device knows if the addresses being read by consecutive threads are consecutive or separated by a constant stride multiple of 32. When the device detects that addresses being read by consecutive threads are separated by a constant stride multiple of 32 (or zero), the program coalesces multiple memory accesses in the same clock cycle, thus allowing to reduce the number of memory transactions.

- *Thread Divergence*: When the kernel runs, consecutive threads are grouped for execution in groups of 32 threads (a warp). When a divergent condition is present (such as if or case statements), the warp checks both possible paths of execution, resulting in an additional clock cycle. If all threads follow a different path, execution is serialized. Thus, whenever possible, all divergent branches should be eliminated.

3.2 Filtering Parallelization

The segmentation and shape-based detector algorithms proposed (Chapter 2.3 and 2.4) make use of the filtering process intensively, which is a time expensive procedure. The filtering procedure has been identified as being responsible for up to 98% of the global processing time (please see [21]), involved in computing the Hessian matrix in the shape-based detector functions (blood detection in Wireless Capsule Endoscopy (WCE) images). So, we studied and implemented several versions of filtering procedure (conventional and separable approaches in time-domain versions and frequency-domain approach running C/C++ code on CPU and GPU). With the obtained results of this study, we select the fastest implementation in time-domain (Separable filtering method) and frequency-domain under CPU and GPU in order to perform more intensively benchmarks using small and large filters. The CUDA separable filtering (Time-Domain) approach running on GPU makes use of global memory, constant memory and shared memory as described below in Algorithm 1.

Algorithm 1 Separable Filtering CUDA Algorithm

1: (**load_image**) Load image to CPU memory
2: (**compute_filter**) Compute 1D filter's on CPU (Rows and Column)
3: (**CPU→GPU memory transfer**) Copy image and filter data to GPU Global memory
4: (**GPU memory transfer**) Copy rows and columns filter data to GPU Constant memory
5: (**GPU memory transfer**) Copy image data to GPU Shared memory
6: (**filter_rows**) Filter image rows with row filter, each thread computes just one pixel from reading N(filter lenght) neighbor pixels of image by sharing memory between threads of the same block
7: (**GPU memory transfer**) Store results on buffer in GPU global memory
8: (**GPU memory transfer**) Copy buffer from GPU global memory to GPU shared memory
9: (**filter_columns**) Filter buffer columns with column filter, each thread computes just one pixel from reading N(filter length) neighbor pixels of buffer by sharing memory between threads of the same block
10: (**GPU memory transfer**) Store filtering results in Global memory
11: (**GPU→CPU memory transfer**) Copy filtering results to CPU memory

The CUDA Fast Fourier Transform (FFT) 2D filtering (Frequency-Domain) approach running on GPU makes use of global memory as described below in Algorithm 2.

Algorithm 2 FFT 2D Filtering CUDA Algorithm

1: (**load_image**) Load image to CPU memory
2: (**compute_filter**) Compute 2D filter on CPU
3: (**CPU→GPU memory transfer**) Copy image data and filter data to GPU Global memory
4: (**compute_FFT2D**) Convert image and filter to (Frequency-Domain) using "cuFFT" FFT2D
5: (**multiply"filter"**) Perform the point-wise multiplication of the FFT of image and filter (Complex Number Multiplication), each thread processes just one pixel, reading one entry from image and one entry from filter to perform the two complex number multiplication
6: (**compute_IFFT2D**) Convert result to (Time-Domain) using "cuFFT" IFFT2D
7: (**GPU→CPU memory transfer**) Copy filtering result data to CPU memory

Next, we show filtering benchmarks that are performed under original and resized WCE images. The resized WCE images are 3 times larger than original ones, and thus, the size of objects (blood regions) will be larger, implying the use of filters with higher dimensions.

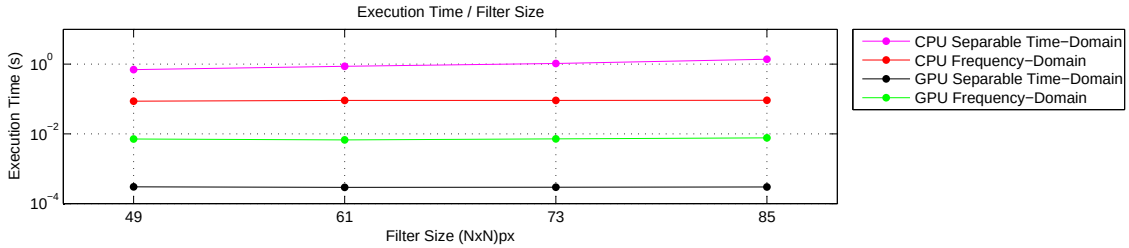


Figure 3.4: Execution times for filtering procedure applied to images with 576×576 pixel varying the processing platform and filter size: 49×49; 61×61; 73×73 and 85×85. The tests were performed on WCE images executing on NVidia Geforce GTX 680 GPU, applying 3 filters for each dimension.

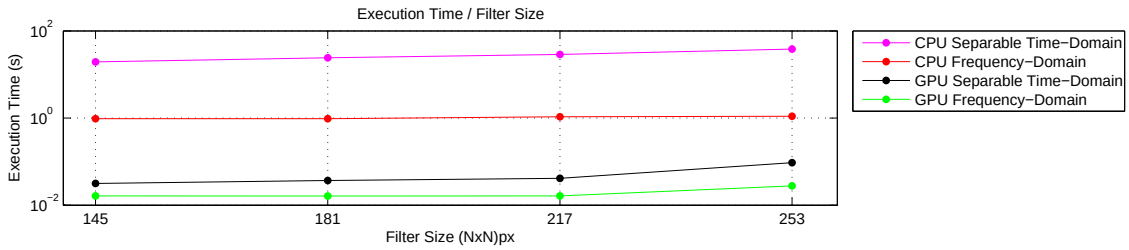


Figure 3.5: Execution times for filtering procedure applied to images with 1728×1728 pixel varying the processing platform and filter size: 145×145; 181×181; 217×217 and 253×253. The tests were performed on resized WCE images 3 times larger than the original size executing on NVidia Geforce GTX 680 GPU, applying 3 filters for each dimension.

Thus, through Figures 3.4 and 3.5 we can conclude that separable time-domain approaches executing on GPU are faster for small filters and that frequency-domain approaches running on GPU are more efficient for larger filters (see [21] for more information). In CPU implementations, for the filter dimensions tested, the filtering process is always faster using the frequency-domain method with the FFTW3.3.3 library [20].

3.2.1 Principles Behind Separable Time-Domain Filtering

Filtering procedure has been proved to be a useful method in image processing, but it's a time expensive task [21]. To perform a 2D filtering in time-domain using a filter with width x, it takes x^2wh multiplications and additions (using an image with $w \times h$ size). However, some 2D filtering filters used in image processing can be broken down

to two 1D filters. Through these two filters, we can perform a horizontal and vertical filtering, with lower complexity: only $2kwh$ multiplications and additions are needed. With this separable filtering method, the obtained results are the same of obtained through conventional 2D filtering [29].

A 2D filter is separable, if the filter K can be expressed as the outer product of two 1D vectors u and v. Take this 3×3 matrix as example:

$$K = v \otimes u = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \otimes \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} = \begin{bmatrix} v_1u_1 & v_1u_2 & v_1u_3 \\ v_2u_2 & v_2u_2 & v_2u_3 \\ v_3u_3 & v_3u_2 & v_3u_3 \end{bmatrix} \tag{3.1}$$

Once these vectors are calculated, the filter is separated: u represents the horizontal, and v the vertical 1D filter. In image processing applications there are several image filters that can be implemented this way (e.g. Averaging Filter; Gaussian Filter; Sobel Operator; Prewitt Operator; and all other filters with rank of the K matrix being 1).

We compute our equivalent separable filters using MATLAB functions which are: Separate Kernel in 1D kernels [30] and Kernel decomposition [31].

3.2.1.A Separable Filtering CUDA

By dividing the image into parts each to be processed in a block, it is possible to ensure that each thread block has a properly aligned base address for coalesced memory accesses. But, in filtering procedure there is a portion of pixels around the image block within a thread block, called by halo region. This halo region of pixels has the same width of filter radius and is required in order to filter the image block. Thus, each thread block must load into shared memory the image pixels to be filtered and the image halo pixels as well. Normally, the threads with horizontal index 0 ($threadIdx.x == 0$) load all the leftmost halo pixels, thus memory accesses won't be aligned and we will produce coalesced memory accesses. The solution to this problem is presented in Figure 3.6. Using a small extra amount of shared memory, we can ensure that memory accesses of the threads are properly aligned: the entire block loads both the left and right halo pixels inside the block width. These redundant memory loads will not represent a penalty and the load of the entire halo region will be coalesced and performed in a single clock cycle. With this, each thread loads the same number of pixels to the shared memory and no branching is necessary to check if the thread is inside the halo.

The vertical filter uses a similar approach, but this time with the thread index increasing perpendicularly to the filter direction rather than along it. Figure 3.7 depicts the layout of memory accesses in the vertical filtering pass. To match coalescing requirements, it is preferable to set the width to 32 (whenever possible, but with large filters, we can not do it due to the size limitations of shared memory), so that each row of the block can be loaded

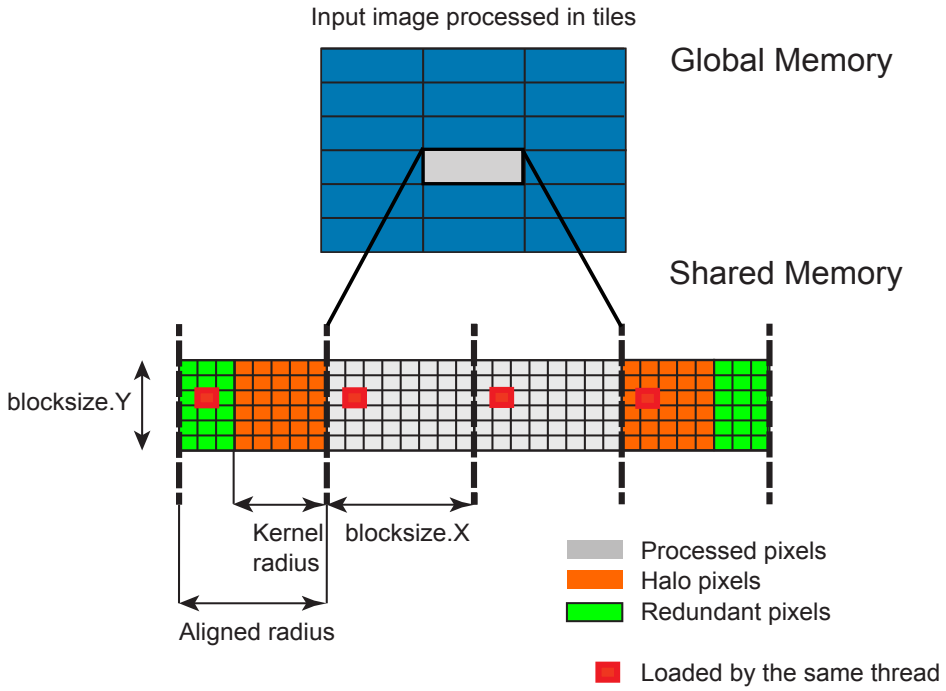


Figure 3.6: Horizontal filter pass processed by a single thread block. Here, a single thread can load and process multiple pixels in the horizontal direction.

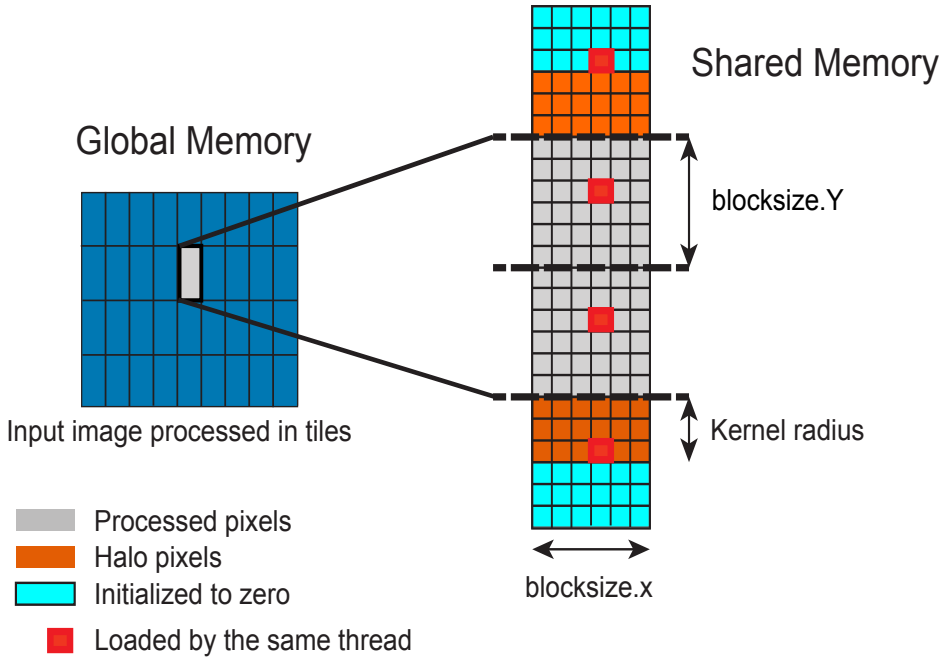


Figure 3.7: In the vertical pass, the concept of implementation is the same as in the horizontal pass, but note that the redundant pixels in the shared memory are initialized to zero this time (not loaded from global memory), as loading additional pixels would imply redundant memory transactions as well.

in a single clock cycle. As the horizontal pass, here each thread loads multiple elements to the shared memory, reducing the number of overlapped pixels of different tiles. A full description of 2D separable filtering using CUDA can be found in [28].

3.3 Blood detector parallelization

The blood detection algorithm is composed by two steps: segmentation and shape-based detection (described in chapters 2.3 and 2.4). In this chapter we present a description of blood detection parallelization.

3.3.1 Segmentation parallelization

In Figure 3.8, we depict the segmentation pipeline executed on GPU, where in all steps data is computed in parallel.

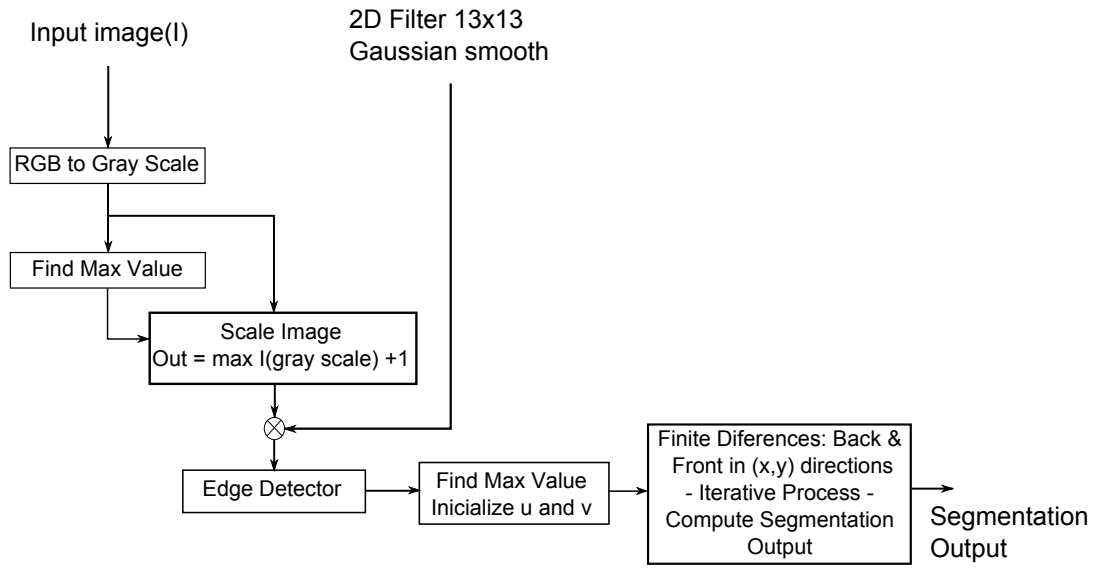


Figure 3.8: Segmentation pipeline processed on the GPU.

Some operations in the segmentation process, referred in Chapter 2.3, need to share image data between threads. In order to achieve higher processing speed, shared memory should be used, whenever possible as mentioned before (see [32] for a related work). The developed functions capable of exploiting shared memory are: finding maximum and mean values; and 2D separable filtering [28]. All other functions perform slower if shared memory is used, since the total number of transactions with global memory introduces higher penalties.

The results of maximum and mean values are processed in two stages: the first stage uses GPU grids with 256×256 block size; and the second one uses 1×256. In the 2D

filtering, block sizes of dimension 16×16 are used.

The remaining functions in the segmentation process only use global memory and GPU grids with 1296×256 block sizes as best performing configurations.

3.3.2 Detector parallelization

Figure 3.9 depicts the detector pipeline executed on the GPU, where again in all steps data is computed in parallel.

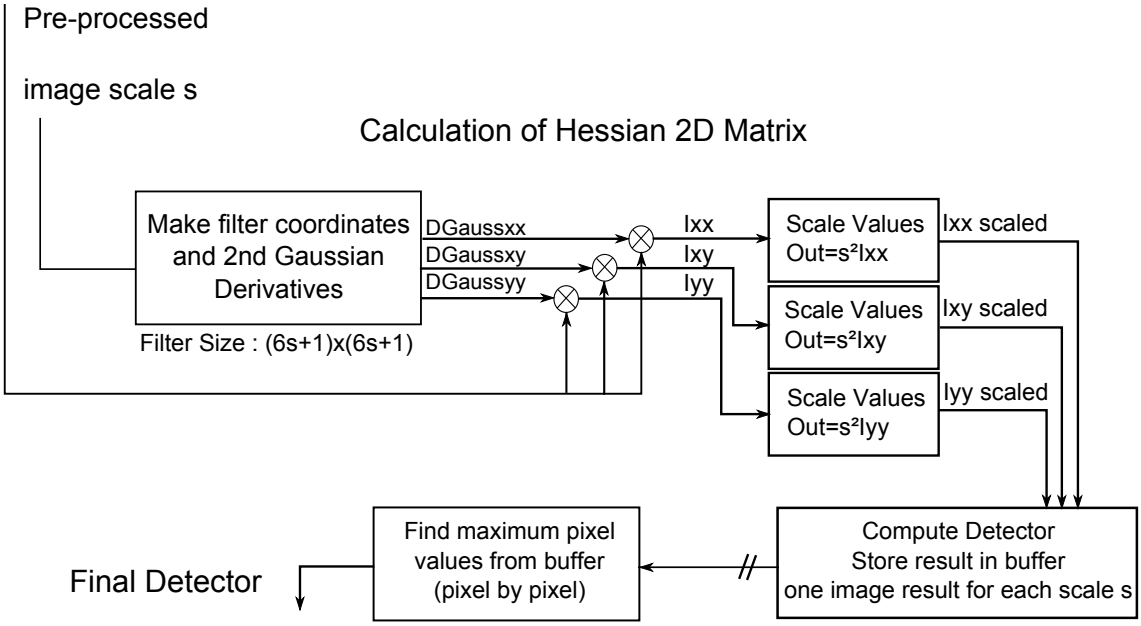


Figure 3.9: Detection pipeline processed on the GPU.

To accelerate the detection of blood described in Chapter 2.4 (Shape-Based Detector), only one operation shares image data between threads: 2D separable filtering [28]. If we use shared memory to perform the remaining functions we observe slower throughput, since the total number of data exchanges with global memory represents a penalizing impact. The results of 2D separable filtering are computed using block sizes of dimension 16×16 and 8×8 for the scale values $s = [8\ 10]$ and $s = [12\ 14]$ (see Chapter 2.4), respectively. All other functions always use global memory with 8×8 block size.

3.4 Exudates detection parallelization

For speeding up the Exudates detection procedure, described in Chapter 2.4 (Shape-Based Detector), we only use one function that shares image data between threads: 2D separable filtering [28]. Again, the remaining functions perform slower if we use shared memory because the total number of transactions to global memory would assume a

higher impact. The results of 2D separable filtering are computed using block sizes of dimension 8×8 for the scale values $s = [2\ 5\ 8\ 11\ 14]$ (see Chapter 2.4). All other functions always use global memory blocks with size 8×8.

3.5 Hybrid GPU-GPU Computing

To achieve higher throughput performance we propose to use a multi-GPU approach. In this chapter, a suitable hybrid GPU-GPU framework is proposed for speeding up the segmentation and shape-based detection procedures. In hybrid GPU-GPU assemblies, we need to find a suitable load balance between resources, deciding a priori how many images we can process on each GPU within the same approximate period of time. This decision is performed by a training process, which checks how many CUDA devices are available on the machine and runs several times the single-GPU versions (segmentation and blood detector in WCE images or exudates detection in retinal fundus images) for each GPU. For all CUDA devices, the average execution times in single-GPU assemblies allows finding the ratio of performance between all GPUs. This training process just needs to be run once (unless hardware changes have been made, more specifically if any CUDA device has changed), in order to build a simple configuration file. Our hybrid GPU-GPU algorithm automatically adapts to the available resources (CUDA devices), which means it is 100% portable across different machines who support CUDA, thats only need to run the training process first.

We propose two multi-GPU assemblies. In the first hybrid GPU-GPU assembly (NVidia GTX 680, NVidia Tesla C1060 and C2050), the training process has shown that we should process a different number of image per GPU. The values obtained in the training process are different for segmentation and shape-based detection. The obtained values: for segmentation are [3,1,2]; and for shape-based detection are [3,1,3], respectively. The different results in the training process between segmentation and shape-based detection are caused by small performance differences between all CUDA devices, some of which are faster than others regarding to data transfers (global and shared memory), while others have higher compute capabilities, namely regarding to mathematical calculations. As expected, in the second hybrid GPU-GPU assembly (Dual GPU NVidia GTX TITAN), the training process has shown a $50-50\%$ workload balance on each GPU, since we use the same GPU model with equivalente computational power. Figures 3.10 and 3.11 illustrate the execution profile in all CUDA devices. Through these figures it is shown that multiple tasks are performed concurrently, such as kernels execution and data transfers. By using these multi-GPU assemblies it is possible to process multiple frames at the same time in different devices.

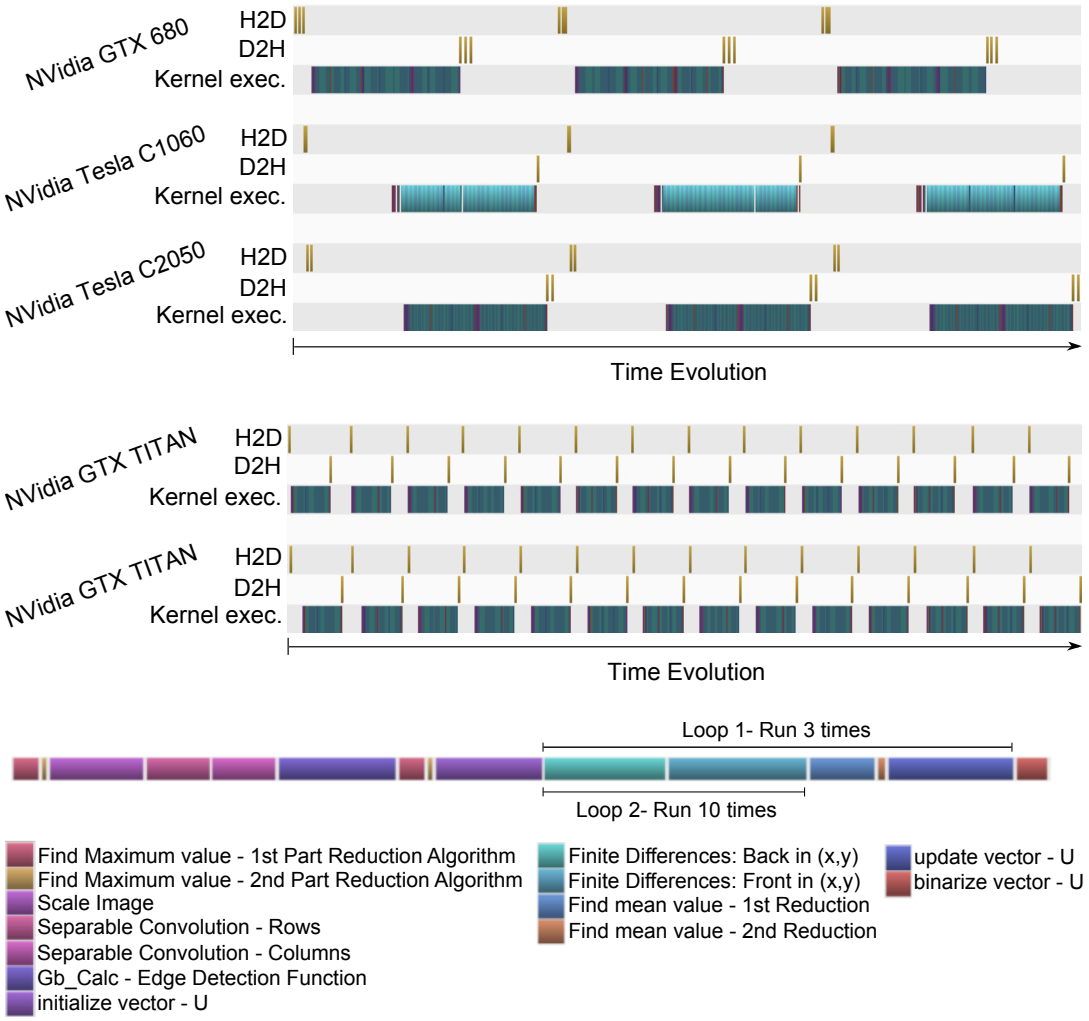


Figure 3.10: Execution pipeline of segmentation procedure on hybrid GPU-GPU assemblies. a) Assembly 1 (GPU NVidia Tesla C1060; GPU NVidia Tesla C2050; GPU NVidia GTX 680) b) Assembly 2 (Dual GPU NVidia GTX TITAN) c) Kernel execution order to process an image. In the segmentation procedure, we identify a short segment which include two different loops: Loop 1 and Loop 2. Loop 1 is responsible to call Loop 2, find mean values and updating the fitting term (see chapter 2.3), this loop runs 3 times (see Figure 3.8. Loop 2 is responsible to compute finite differences method: Back and Front and update vector u and v, this loop runs 10 times for every call (with this, Loop 2 will run 30 times).

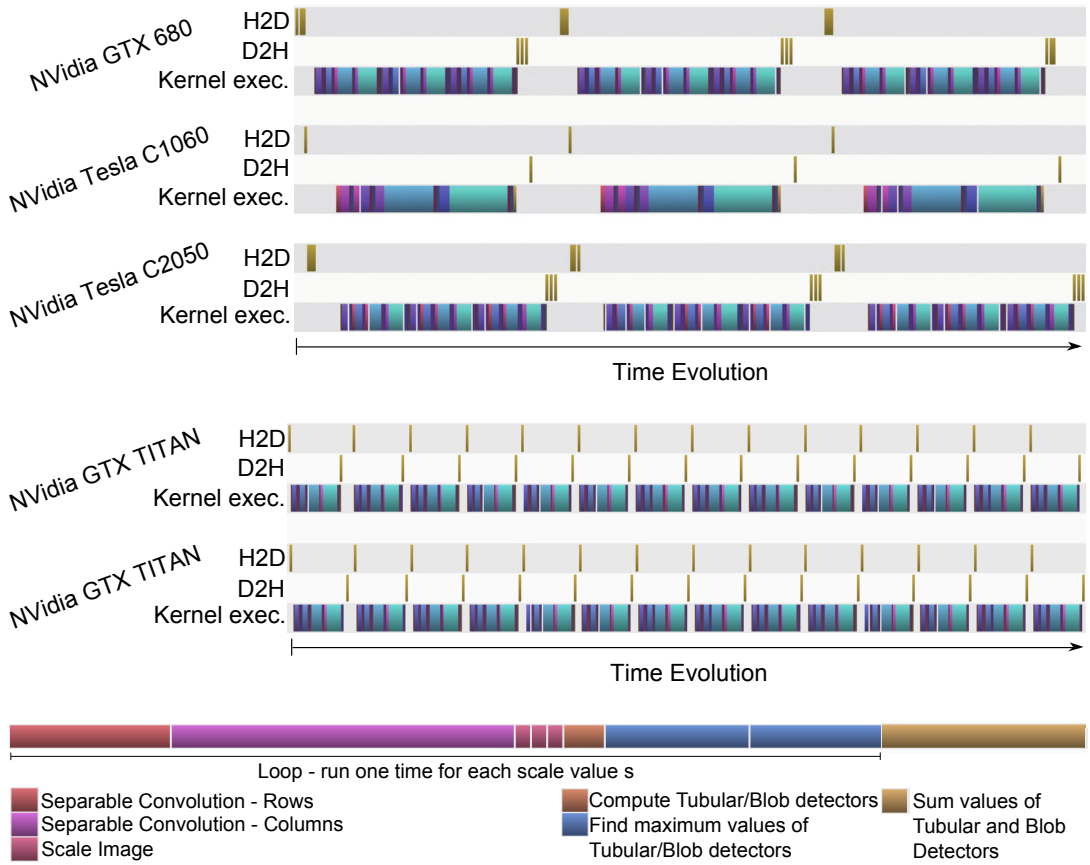


Figure 3.11: Execution pipeline of shape-based detection procedure on hybrid GPU-GPU assemblies. For blood detection $s = 8, 10, 12, 14$ and $s = 2, 5, 8, 11, 14$ for exudates detector. a) Assembly 1 (GPU NVidia Tesla C1060; GPU NVidia Tesla C2050; GPU NVidia GTX 680) b) Assembly 2 (Dual GPU NVidia GTX TITAN) c) Kernel execution order to process an image. In the shape-based detector procedure, we identify a short segment which includes a loop. This loop runs one time for each scale value s, in order to compute Hessian matrix (see equation 2.8) and detector functions (please see equation 2.11 and 2.12) for s value (for blood detector this code runs 4 times and 5 times to compute exudates detector).

4

Experimental Results and Speedup

Contents

4.1 Results Using Single-GPU Systems

In this chapter, the results and speedup obtained by applying parallelization techniques to the two case studies are presented.

4.1.1 Blood detection results

The speedups obtained in the blood detection algorithm, are separated into two steps: segmentation and shape-based detection.

4.1.1.A Segmentation results

The computation times regarding the segmentation procedure are presented in Table 4.1, that shows the frames-per-second (FPS) obtained using parallel computation on single-Graphics Processing Unit (GPU) systems. As displayed, this procedure runs 68 times faster on GPU NVidia GTX TITAN (processing one image in 3.5ms), when compared to an Intel i7 Central Processing Unit (CPU).

Processing Platform	Segmentation execution time (ms)	FPS
CPU Intel i7	240.0	4.2
GPU NVidia Tesla C1060	12.91	77.46
GPU NVidia Tesla C2050	5.84	171.23
GPU NVidia GTX 680	4.81	207.90
GPU NVidia GTX TITAN	3.5	285.71

Table 4.1: Computation times in milliseconds (ms) for the segmentation procedure and throughput measured in FPS. All tests were performed on WCE images with 576×576 pixels.

4.1.1.B Detection results

Processing Platform	Detector execution time (ms)	FPS
CPU Intel i7	529.9	1.9
GPU NVidia Tesla C1060	17.78	56.24
GPU NVidia GTX 680	7.00	142.86
GPU NVidia Tesla C2050	6.65	150.38
GPU NVidia GTX TITAN	4.84	206.61

Table 4.2: Computation times in millisecons (ms) for the blood detector function and throughput measured in FPS. All tests were performed on WCE images with 576×576 pixels.

The blood detector computation times are presented in Table 4.2. This algorithm runs 109 times faster on GPU NVidia GTX TITAN (processing one image in 4.84ms), when compared to an Intel i7 CPU.

4.1.1.C Complete blood detection procedure - Global Speedup

Table 4.3 shows throughput measured in FPS and the corresponding speedups achieved performing the complete blood detection algorithm (segmentation and shape-based detector).

Processing Platform	Segmentation and Detector (fps)	Speedup
GPU NVidia Tesla C1060	32.58	25.06x
GPU NVidia Tesla C2050	80.06	61.58x
GPU NVidia GTX 680	84.67	65.13x
GPU NVidia GTX TITAN	119.90	92.23x

Table 4.3: Throughput measured in FPS and speedup archived to the complete blood detector algorithm (Segmentation and Blood Detector) comparing against a sequential version running on an Intel i7 CPU. Tests performed on WCE images with 576×576 pixels.

The blood detection procedure runs 92 times faster on GPU NVidia GTX TITAN (fastest single-GPU version). With the obtained speedup, the GPU NVidia GTX TITAN shows to be able of processing 119.90 FPS, which is equivalent to observe that the approximate total number of 56000 frames, generated by a complete WCE exam, can be computed in less than 8 minutes.

Processing Platform	GPU Cost($)	Cost($)/FPS
GPU NVidia GTX 680	291.95	3.45
GPU NVidia Tesla C1060	199.95	6.14
GPU NVidia GTX TITAN	1078.73	9.00
GPU NVidia Tesla C2050	750	9.37

Table 4.4: Cost per Processed frame per second in blood detector algorithm (Segmentation and Blood Detector). Tests performed on WCE images with 576×576 pixels.

In Table 4.4 we show the current GPUs cost, and depict a relation between the cost and throughput measured in FPS. By analyzing this table we can see that for a simple blood detection in real-time, like a common endoscopy, the NVidia GTX 680 GPU can perform *in vivo* diagnosis. With an inexpensive NVidia GTX 680 GPU card ($291.95) we are able to process 84.67 FPS.

4.1.2 Exudates detection results

The computation times of the exudates detector procedure are presented in Table 4.5, showing time in millisecons (ms) and throughput measured in FPS.

Through Table 4.6, we clearly see the speedup obtained using parallel computation on GPU. This algorithm runs 179.45 times faster on GPU NVidia GTX TITAN, when compared to an Intel i7 CPU. With this obtained speedup, the NVidia GTX TITAN GPU shows to be able of processing 16.73 FPS.

Processing Platform	Detector execution time (ms)	FPS
CPU Intel i7	10725.5	0.093
GPU NVidia Tesla C1060	236.52	4.23
GPU NVidia GTX 680	86.85	11.51
GPU NVidia Tesla C2050	81.97	12.2
GPU NVidia GTX TITAN	59.77	16.73

Table 4.5: Computation times in millisecons (ms) for the exudates detector procedure and throughput measured in FPS. The tests were performed on HD RF images with 2416×1736 pixels.

Processing Platform	Speedup
GPU NVidia Tesla C1060	45.35x
GPU NVidia GTX 680	123.49x
GPU NVidia Tesla C2050	130.85x
GPU NVidia GTX TITAN	179.45x

Table 4.6: Speedup obtained to the exudates detection, when compared against a sequential version running on an Intel i7 CPU. Tests performed on HD RF images with 2416×1736 pixels.

In Table 4.7 we present the current GPU's cost, and show a relation between cost and throughput measured in FPS. The analisys of this table allows to conclude that the NVidia GTX 680 GPU is the most inexpensive, considering cost per frame.

Processing Platform	GPU Cost($)	Cost($)/FPS
GPU NVidia GTX 680	291.95	25.36
GPU NVidia Tesla C1060	199.95	47.30
GPU NVidia Tesla C2050	750	61.48
GPU NVidia GTX TITAN	1078.73	64.48

Table 4.7: Cost per processed frame per second in exudates detector algorithm. Tests performed on HD retinal fundus images with 2416×1736 pixels.

4.2 Results Using Multi-GPU Systems

We present bellow the results and speedup obtained by applying parallelization techniques in both case studies, which are: detection of blood in WCE images and detection of exudates in HD RF images, using hybrid GPU-GPU systems.

4.2.1 Blood detection results

The speedup obtained in the blood detection algorithm, is separated into two steps: segmentation and shape-based detection.

4.2.1.A Segmentation results

Processing Platform	Segmentation execution time (ms)	FPS
CPU Intel i7	240.0	4.2
GPU C1060 + C2050 + GTX680	3.22	310.56
GPU Dual GTX TITAN	1.98	505.05

Table 4.8: Computation times in milliseconds (ms) for the segmentation procedure and throughput measured in FPS. The tests were performed on WCE images with 576×576 pixels.

The computation times regarding the segmentation procedure are presented in Table 4.8, that shows the FPS obtained using parallel computation on hybrid GPU-GPU systems. As displayed, this procedure runs 121 times faster on DUAL NVidia GTX TITAN, when compared to an Intel i7 CPU. With this fastest hybrid GPU-GPU approach, we can process 1.77 times faster than best single-GPU version (NVidia GTX TITAN).

4.2.1.B Detection results

The computation times of the blood detector function are presented in Table 4.9. This algorithm runs 204 times faster on DUAL NVidia GTX TITAN, when compared to an Intel i7 CPU. With this fastest hybrid GPU-GPU approach, we can process 1.87 times faster than best single-GPU version (NVidia GTX TITAN).

4.2.1.C Complete blood detection procedure - Global Speedup

Table 4.10 shows throughput measured in FPS and the speedup achieved in full blood detector algorithm (Segmentation and Shape-Based Detector).

Processing Platform	Detector execution time (ms)	FPS
CPU Intel i7	529.9	1.9
GPU C1060 + C2050 + GTX680	3.51	284.90
GPU Dual GTX TITAN	2.59	386.10

Table 4.9: Computation times in millisecons (ms) for the blood detector function and throughput measured in FPS. The tests were performed on WCE images with 576×576 pixels.

Processing Platform	Segmentation and Detector (fps)	Speedup
GPU C1060 + C2050 + GTX680	148.59	114.30x
GPU Dual GTX TITAN	218.82	168.32x

Table 4.10: Throughput measured in FPS and speedup archived to the complete blood detector algorithm (Segmentation and Blood Detector) comparing against a sequential version running on an Intel i7 CPU. Tests performed on WCE images with 576×576 pixels.

The blood detection procedure runs 168 times faster on DUAL NVidia GTX TITAN, when compared to an Intel i7 CPU. With this fastest hybrid GPU-GPU approach, we can process 1.83 times faster than best single-GPU version (NVidia GTX TITAN).

Thus, the Dual NVidia GTX TITAN assembly shows to be able of processing 218.82 FPS, which is equivalent to observe that the approximate total number of 56000 frames, generated by a complete WCE exam, can be processed in less than 5 minutes.

Processing Platform	GPU Cost($)	Cost($)/FPS
GPU Dual GTX TITAN	2157.46	9.86
GPU C1060 + C2050 + GTX680	1941.9	13.07

Table 4.11: Cost per Processed FPS in blood detector algorithm (Segmentation and Blood Detector). Tests performed on WCE images with 576×576 pixels.

In Table 4.11 we show the current GPUs cost, and depict a relation between cost and throughput measured in FPS. Through this table, we can see that the Dual GTX TITAN assembly is the most inexpensive regarding cost per frame.

4.2.2 Exudates detection results

The computation times of the exudates detector procedure are presented in Table 4.12 and the achieved speedup is shown in Table 4.13. This algorithm runs 324.23 times faster on a hybrid GPU-GPU configuration with DUAL NVidia GTX TITAN, when compared to an Intel i7 CPU.

Processing Platform	Detector execution time (ms)	FPS
CPU Intel i7	10725.5	0.093
GPU C1060 + C2050 + GTX680	41.12	24.32
GPU Dual GTX TITAN	33.08	30.23

Table 4.12: Computation times in millisecons (ms) for the exudates detector procedure and throughput measured in FPS. The tests were performed on HD RF images with 2416×1736 pixels.

Processing Platform	Speedup
GPU C1060 + C2050 + GTX680	260.83x
GPU Dual GTX TITAN	324.23x

Table 4.13: Speedup obtained to the exudates detector algorithm (Retinal Fundus Images) comparing against a sequential version running on an Intel i7 CPU. Tests performed on HD RF images with 2416×1736 pixels.

With the obtained speedup, the hybrid GPU-GPU configuration with DUAL NVidia GTX TITAN shows to be able of processing 30.23 FPS, which allows to perform *in vivo* diagnosis.

Processing Platform	GPU Cost($)	Cost($)/FPS
GPU Dual GTX TITAN	2157.46	71.37
GPU C1060 + C2050 + GTX680	1941.9	79.85

Table 4.14: Cost per processed frame per second in exudates detector algorithm. Tests performed on HD RF images with 2416×1736 pixels.

In Table 4.14 we show the current GPU's cost, and point a relation between cost and throughput measured in FPS. Through this table, we verify that the Dual GTX TITAN assembly is the most inexpensive, considering cost per frame.

5

Conclusions

Contents

With the rapidly enhancing performances of Graphics Processing Units (GPUs), improved programming support, and excellent price-to-performance ratio, GPUs have emerged as competitive parallel computing platforms for computationally expensive and demanding tasks in a wide range of medical imaging applications. A GPU-based framework for image segmentation (designed to remove uninformative regions such as bubbles, trash, dark regions, which can interfere with the detection of blood in Wireless Capsule Endoscopy (WCE) images) and shape-based detection are proposed. The core of the shape-based detection algorithm lies in the definition of a good discriminator for detecting blob-like and tubular-like shapes in several applications in image processing. This is accomplished by choosing a suitable color channel, image Hessian eigenvalue analysis and multiscale image analysis approach. This segmentation and shape-based detection is applied in blood detection in WCE images. For the Exudates (EXs) detection on Retinal Fundus (RF) images only the shape-based detector needs to be applied. It should be considered that different configurations were adopted for the two case studies. Experimental results for our current dataset show that the proposed algorithm is effective and achieves 89.56% accuracy when applied to WCE images [1].

Finally, the computation results and speedups are shown in Chapter 4. Tables 4.1, 4.2 and 4.3 show that the GPU NVidia GTX TITAN is the faster single-GPU assembly, with a throughput of 285.71 frames-per-second (FPS) on segmentation and 206.61 FPS on blood detection. With the achieved speedup on single-GPU assemblies, the GPU NVidia GTX TITAN is able to run the full algorithm 92.23 times faster than in the original sequential Central Processing Unit (CPU) version. The proposed hybrid GPU-GPU system with Dual GPU NVidia GTX TITAN achieves a speedup 168.32 faster than the original CPU version and shows to be capable of processing 218.82 FPS, which is equivalent to observe that the approximate total number of 56000 frames, generated by a complete WCE exam, can be computed in less than 5 minutes (please see Tables 4.8, 4.9 and 4.10). In High Definition (HD) RF images we only apply shape-based detection, and the fastest single-GPU system can process 16 frames per second with an average speedup of 179 times compared to the sequential CPU version (please see Tables 4.5 and 4.6). In the proposed hybrid GPU-GPU system we can process 30 frames per second with an average speedup 324 times faster than the original CPU version (please see Tables 4.12 and 4.13).

With these throughputs we are able to build real-time systems to automatically detect exudates lesions in the retina fundus and blood in endoscopic images, which may help the medical practitioner improving the diagnosis procedure. The best hybrid GPU-GPU approach is capable to process blood and exudates detection procedures 1.83 and 1.81 times faster, respectively, than with the best single-GPU version (NVidia GTX TITAN).

These speedups are achieved by parallelizing two crucial steps, segmentation and

blood detector functionalities in the algorithm, that were consuming most of the global processing time. To perform these steps more efficiently we run parallel code on GPUs with an appropriate use of memory (shared and global). This novel approach allows processing multiple images and multiple pixels of an image at the same time, thus sustaining the obtained throughput levels.

5.1 Future Work

In the future, we expect to apply this procedure to an online real video feed, benchmarking the algorithm in real world contexts under medical utilization and supervision, namely with real patients. Another interesting topic of research consists of applying these procedures to low-power and also low-energy consumption devices, that may allow the development of portable devices in the future.

Bibliography

[1] I. N. Figueiredo, S. Kumar, C. Leal, and P. N. Figueiredo, "An automatic blood detection algorithm for wireless capsule endoscopy images," in *Computational Vision and Medical Image Processing,VIPIMAGE* 2013*, João Tavares & Natal Jorge (eds),* 2014 *Taylor & Francis Group, London, ISBN 978-1-138-00081-0 (ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing)*, pp. 237–241.

[2] M. U. Akram, A. Tariq, S. A. Khan, and M. Y. Javed, "Automated detection of exudates and macula for grading of diabetic macular edema," *Computer Methods and Programs in Biomedicine*, pp. 141–152, 2014.

[3] G. Idan, G. Meron, and A. Glukhovsky, "Wireless capsule endoscopy," *Nature*, vol. 405, pp. 417–417, 2000.

[4] M. Bashar, T. Kitasaka, Y. Suenaga, Y. Mekada, and K. Mori, "Automatic detection of informative frames from wireless capsule endoscopy images," *Medical Image Analysis*, vol. 14, pp. 449–470, 2010.

[5] M. Coimbra and J. Cunha, "MPEG-7 visual descriptors-contributions for automated feature extraction in capsule endoscopy," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 628–637, 2006.

[6] L. Cui, C. Hu, Y. Zou, and M. Q.-H. Meng, "Bleeding detetction in wireless capsule endoscopy images by support vector classifier," in *Proceedings of the 2010 IEEE Conference on Information and Automation*, Harbin, China, June 2010, pp. 1746–1751.

[7] J. P. S. Cunha, M. Coimbra, P. Campos, and J. M. Soares, "Automated topographic segmentation and transit time estimation in endoscopic capsule exams," *IEEE Transactions on Medical Imaging*, vol. 27, pp. 19–27, 2008.

[8] B. Li and M. Q.-H-Meng, "Computer-aided detection of bleeding regions for capsule endoscopy images," *IEEE Transactions on Biomedical Engineering*, vol. 56, pp. 1032–1039, 2009.

[9] G. Pan, F. Xu, and J. Chen, "A novel algorithm for color similarity measurement and the application for bleeding detection in WCE," *I.J. Image, Graphics and Signal Processing*, vol. 5, pp. 1–7, 2011.

[10] B. Penna, T. Tilloy, M. Grangettoz, E. Magli, and G. Olmo, "A technique for blood detection in wireless capsule endoscopy images," in *17th European Signal Processing Conference (EUSIPCO 2009)*, 2009, pp. 1864–1868.

[11] M. Liedlgruber and A. Uhl, "Computer-aided decision support systems for endoscopy in the gastrointestinal tract: a review," *IEEE Reviews in Biomedical Engineering*, vol. 4, pp. 73–88, 2011.

[12] I. N. Figueiredo, S. Kumar, C. Leal, and P. N. Figueiredo, "Computer-assisted bleeding detection in wireless capsule endoscopy images," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 1, pp. 198–210, 2013.

[13] I. N. Figueiredo, S. Kumar, and P. N. Figueiredo, "An intelligent system for polyp detection in wireless capsule endoscopy images," in *Computational Vision and Medical Image Processing IV: VIPIMAGE 2013, ISBN: 9781315812922*, Madeira Island, Funchal, Portugal, 2013, pp. 229–235.

[14] Y.-I. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," *Computer Graphics and Image Processing*, vol. 13, pp. 222–241, 1980.

[15] M. Usman Akram, S. Khalid, A. Tariq, S. A. Khan, and F. Azam, "Detection and classification of retinal lesions for grading of diabetic retinopathy," *Computers in biology and medicine*, vol. 45, pp. 161–71, 2014.

[16] X. Zhang, G. Thibault, D. E., M. gui B., L. B., D. R., C. G., Q. G., L. M., M. P., C. A., V. Z., and E. A., "Exudate detection in color retinal images for mass screening of diabetic retinopathy," *Medical Image Analysis*, 2014.

[17] J. Cheng, J. Liu, Y. Xu, F. Yin, D. Wong, N.-M. Tan, D. Tao, C.-Y. Cheng, T. Aung, and T. Y. Wong, "Superpixel classification based optic disc and optic cup segmentation for glaucoma screening," *Medical Imaging, IEEE Transactions on*, vol. 32, no. 6, pp. 1019–1032, June 2013.

[18] M. Krause, R. Alles, B. Burgeth, and J. Weickert, "Fast retinal vessel analysis," *Journal of Real-Time Image Processing*, pp. 1–10, 2013.

[19] V. Podlozhnyuk, M. Harris, and E. Young, "NVIDIA CUDA C programming guide." NVIDIA Corporation, 2012.

[20] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

[21] C. Graca, G. Falcao, S. Kumar, and I. N. Figueiredo, "Cooperative use of parallel processing with time or frequency-domain filtering for shape recognition," in *EUSIPCO 2014 (22nd European Signal Processing Conference 2014) (EUSIPCO 2014)*, Lisbon, Portugal, Sep. 2014.

[22] S. Kumar, I. N. Figueiredo, C. Graca, and G. Falcao, "A gpu accelerated algorithm for blood detection in wireless capsule endoscopy images," in *Tavares and J.M. and Renato and R.S.N.J. (eds) Developments in Medical Image Processing and Computational Vision. Lecture Notes in Computational Vision and Biomechanics*. Springer, 2014.

[23] Y. Zheng, J. Yu, S. B. Kang, S. Lin, and C. Kambhamettu, "Single-image vignetting correction using radial gradient symmetry," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, Los Alamitos, Calif., USA, June 2008, pp. 1–8.

[24] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher, "Fast global minimization of the active contour/snake model," *J Math. Imaging Vis.*, vol. 28, pp. 151–167, 2007.

[25] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Processing*, vol. 10, pp. 266–277, 2001.

[26] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Multiscale vessel enhancement filtering," in *Medical Image Computing and Computer-Assisted Intervention*, Cambridge, MA, USA, 1998, pp. 130–137.

[27] I. N. Figueiredo and S. Kumar, "Wavelet-based computer-aided detection of bright lesions in retinal fundus images," in *Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications*, ser. Lecture Notes in Computer Science, Y. Zhang and J. Tavares, Eds., 2014, vol. 8641, pp. 234–240.

[28] H. Lee, M. Harris, E. Young, and V. Podlozhnyuk, "Image convolution with CUDA." NVIDIA Corporation, 2007.

[29] S. E. (2006), "Separable convolution," in *MATLAB Central File Exchange. Retrieved May 18, 2006.*

[30] D.-J. K. (2010), "Separate kernel in 1d kernels," in *MATLAB Central File Exchange. Retrieved May 18, 2006.*

[31] C. L. (2010), "Kernel decomposition," in *MATLAB Central File Exchange. Retrieved May 18, 2006.*

[32] M. Martins, G. Falcao, and I. N. Figueiredo, "Fast aberrant crypt foci segmentation on the GPU," in *ICASSP'13: Proceedings of the 36th IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2013.

Appendix A

COOPERATIVE USE OF PARALLEL PROCESSING WITH TIME OR FREQUENCY-DOMAIN FILTERING FOR SHAPE RECOGNITION

Carlos Graca[⋆] *Gabriel Falcao*[⋆] *Sunil Kumar*[†] *Isabel N. Figueiredo*[†]

[⋆] Instituto de Telecomunicações, Dept. of Electrical and Computer Eng., University of Coimbra, Portugal
[†]CMUC, Dept. of Mathematics, University of Coimbra, Portugal

ABSTRACT

For many computer vision applications, detection of blobs and/or tubular structures in images are of great importance. In this paper, we have developed a parallel signal processing framework for speeding up the detection of blob and tubular objects in images. We identified filtering procedure as being responsible for up to 98% of the global processing time, in the used blob or tubular detector functions. We show that after a certain dimension of the filter it is beneficial to combine frequency-domain techniques with parallel processing to develop faster signal processing algorithms. The proposed framework is applied to medical wireless capsule endoscopy (WCE) images, where blob and/or tubular detectors are useful in distinguishing between abnormal and normal images.

Index Terms— Object shape recognition, Convolution, Frequency-domain filtering, Parallel processing, Wireless capsule endoscopy

1. INTRODUCTION

In the field of computer vision, blob/tubular detection refers to methods that are aimed at detecting clustered points in the image that are either brighter or darker than the surrounding. Detection of blob and/or tubular structures in images is an important step in the analysis of a large-scale of scientific data, as for example, detection of bleeding/blood regions in WCE images [1, 2], nodule detection in thorax x-ray images citesch, nuclei detection in microscopic zebrafish images [3], enhancement of vascular structures [4, 5], detection of lesions in images of multiple sclerosis patients [6], and so on.

In this paper it is proposed a parallel signal processing framework which accelerates significantly the performance of particular blob and tubular detectors. These latter rely on the eigenvalues of the Hessian of the processed input image [5], which involve the calculation of second derivatives, at multiple scales, using Gaussian filtering. We identified this filtering process as being responsible for a large part of the global processing time, which turns it a natural candidate for parallelization.

Recently, graphics processing units (GPU) have shown significant speedups for signal processing algorithms in medical imaging areas that require intensive computation [7, 8]. This technology supports specialized parallel kernel development and efficient signal processing libraries that suit well into a variety of biomedical image processing applications.

We propose to cooperatively exploit time- or frequency-domain signal processing techniques combined with more efficient parallel processing. These algorithmic transformations allow producing faster code that can accommodate multi-thread based parallelism with an appropriate use of the system's memory hierarchy and coalesced memory accesses for performing the most compute-intensive procedures on the GPU. We show that depending on the filter size, different domains and architectures can be adopted. For example, for large filters the GPU-based frequency-domain approach obtains higher speedups, while for smaller filters the separable time-domain approach on the GPU performs faster.

Finally, we show an application of the proposed parallel blob and tubular detector algorithm in the medical field, for bleeding/blood detection in WCE images [1, 2]. Note that WCE examination of a patient produces approximately 56000 images. Hence, a major and relevant direct application of the proposed framework is the acceleration of an automated WCE image analysis. In addition, the proposed parallelized procedure can also be incorporated in many other applications (see for instance [3–6, 9]), where blob and/or tubular detectors might be used.

After this short introduction, the rest of the paper is structured as follows. Section 2 introduces the blob and tubular detectors. Section 3 describes the parallelization of blob and tubular detectors. An application to medical images is analyzed in Section 4, and finally some conclusions are given in Section 5.

2. SHAPE-BASED OBJECT RECOGNITION

The definition of the blob and tubular detectors used herein rely on appropriate functions that involve the Hessian eigenvalues of the input image and on a multiscale analysis approach. For a scalar image $I : \Omega \subseteq \mathbb{R}^2 \rightarrow \mathbb{R}$, we define the Hessian matrix at a point (or equivalently at a pixel location)

(x, y), and for a scale s, by

$$H_s(x,y) = \begin{pmatrix} I^s_{xx} & I^s_{xy} \\ I^s_{xy} & I^s_{yy} \end{pmatrix}.$$

Here I^s_{xx}, I^s_{xy} and I^s_{yy} are the second-order partial derivatives of I and the scale s is involved in the calculation of these derivatives using Gaussian filtering. The Hessian matrix describes the second order local image intensity variations around the selected point. Suppose $\lambda_{s,1}$ and $\lambda_{s,2}$ are two eigenvalues of the Hessian matrix H_s. Note that at a point belonging to a blob region, these two eigenvalues have the same sign (the sign is an indicator of the brightness/darkness of the blob: if positive it is a dark blob on a bright background, and if negative it is a bright blob on a dark background) and similar magnitudes. If the point belongs to a tubular structure (like a ridge) one of the eigenvalues is close to zero and the others absolute value is large. Moreover the tubular structure is bright (resp. dark) if the eigenvalue with highest absolute value is negative (resp. positive) (see [5]). Without loss of generality we assume that $|\lambda_{s,1}| \leq |\lambda_{s,2}|$.

Defining now

$$f_1 = \exp\left(-\beta F_s^2\right) \quad \text{and} \quad f_2 = \left(1 - \exp\left(-\alpha \left(\frac{\lambda_{s,1}}{\lambda_{s,2}}\right)^2\right)\right),$$

and motivated from [5], we define the blob (B_s) and tubular (T_s) detectors (at each point of the domain), by

$$B_s = \begin{cases} 0, & \text{if } \lambda_{s,1}\lambda_{s,2} < 0 \text{ or } |\lambda_{s,2} - \lambda_{s,1}| > \delta \\ (1-f_1)f_2, & \text{otherwise,} \end{cases} \tag{1}$$

and

$$T_s = \begin{cases} 0, & \text{if } \lambda_{s,2} > 0, \\ (1-f_1)(1-f_2), & \text{otherwise.} \end{cases} \tag{2}$$

Here α and β are the parameters which control the sensitivity of the functions and δ is the user chosen threshold.

In order to automatically detect blobs (or tubes) of different sizes, a multiscale approach is necessary. The response of the detector functions will be maximum at a scale that approximately matches the size of the structure (blob or tube) to detect. Hence, we define the final detector functions as follows:

$$B = \max_{s_{min} \leq s \leq s_{max}} B_s \quad \text{and} \quad T = \max_{s_{min} \leq s \leq s_{max}} T_s,$$

where s_{min} and s_{max} are the minimum and maximum scales at which the structures are expected to be found.

3. PARALLELIZATION OF THE SHAPE-BASE OBJECT RECOGNITION PROCEDURE

The procedure for detecting blob and tubular structures in images, described in Section 2, is implemented using time- and frequency-domain approaches for calculating the filtering procedure (involved in computing the Hessian matrix), and it is also tested under different execution environments and platforms. The function used to perform the filtering in the time-domain (C/CPU) was released by NVIDIA [10] and in frequency-domain (C/CPU), we used the optimized FFTW3.3.3 library [11]. We find out that the vast majority of time spent processing each image is being consumed by the filtering process (see Tables 1 and 2), which is heavily used for the calculation of the Hessian matrix. As depicted in Fig. 1, three filters are applied for each scale s with a filter size $(6s+1) \times (6s+1)$.

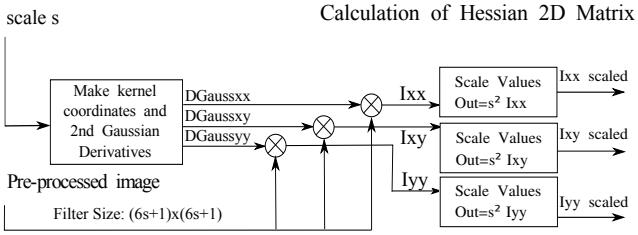


Fig. 1: Block diagram explaining the computation of the Hessian matrix for each scale s (Ixx, Iyy, Ixy are the notations for the second-order partial derivatives of image I).

3.1. GPU parallelization

The parallelization of the procedure for detecting blob and tubular structures in images, described in Section 2, is carried out using the Compute Unified Device Architecture (CUDA) parallel programming model, by exploiting the massive use of thread- and data-parallelism on the graphics processing units (GPU). CUDA allows the programmer to write in a transparent way, scalable parallel C code [12] on GPUs. When the host launches a kernel, the GPU device executes a grid of thread blocks, where each block has a predefined number of threads executing the same code segment.

3.1.1. Parallelization with Separable 2D Filtering

We perform a benchmark of CUDA separable filtering (Time-Domain) [10]. This version uses global memory, constant memory and shared memory as described below in Algorithm 1 .

3.1.2. Parallelization with FFT 2D Filtering

Following a frequency-domain strategy, we also perform a benchmark of CUDA FFT2D filtering (Frequency-Domain) [13]. This version uses global memory as described in Algorithm 2 .

Version & platform	TUBULAR exec. time(s)	Filtering (% of TUB. exec. time)	BLOB exec. time (s)	Filtering (% of BLOB exec. time)
SEP Time-Domain (CPU)	4.0597	94.3720	4.2253	94.5556
Frequency-Domain (CPU)	0.5209	68.4008	0.5299	68.2204
SEP Time-Domain (GPU)	0.2372	0.5011	0.2419	0.4900
Frequency-Domain (GPU)	0.4149	7.4623	0.3773	7.6093

Table 1: Total computation times (in seconds) for TUBULAR and BLOB object-shape detection, and the time percentages (with respect to the total time) of the filtering step, for the different versions and platforms. Tests done with (576×576 pixel) WCE images, applying a total of 12 filters (3 for each dimension) with sizes: 49×49, 61×61, 73×73, 85×85 for TUBULAR test and 49×49, 61×61, 73×73, 97×97 for BLOB test (SEP is the notation for Separable Filtering).

Version & platform	TUBULAR exec. time(s)	Filtering (% of TUB. exec. time)	BLOB exec. time (s)	Filtering (% of BLOB exec. time)
SEP Time-Domain (CPU)	108.1980	98.18	113.221	97.9193
Frequency-Domain (CPU)	5.5730	74.4302	5.411	73.4984
SEP Time-Domain (GPU)	3.1628	6.1323	3.6096	5.66
Frequency-Domain (GPU)	2.2628	2.8915	2.2924	3.3429

Table 2: Total computation times (in seconds) for TUBULAR and BLOB object-shape detection, and the time percentages (with respect to the total time) of the filtering step, for the different versions and platforms. Tests done with (1728×1728 pixel) WCE images, applying a total of 12 filters (3 for each dimension) with sizes: 145×145, 181×181, 217×217, 253×253 for TUBULAR test and 145×145, 181×181, 217×217, 289×289 for BLOB test (SEP is the notation for Separable Filtering).

Algorithm 1 Separable (SEP) Filtering CUDA Algorithm

1: (**load_image**) Load image to CPU memory
2: (**compute_filter**) Compute 1D filter's on CPU (Rows and Column)
3: (**CPU→GPU memory transfer**) Copy image and filter data to GPU Global memory
4: (**GPU memory transfer**) Copy rows and columns filter data to GPU Constant memory
5: (**GPU memory transfer**) Copy image data to GPU Shared memory
6: (**convolve_rows**) Convolve image rows with row filter, each thread computes just one pixel from reading N(filter lenght) neighbor pixels of image by sharing memory between threads of the same block
7: (**GPU memory transfer**) Store results on buffer in GPU global memory
8: (**GPU memory transfer**) Copy buffer from GPU global memory to GPU shared memory
9: (**convolve_columns**) Convolve buffer columns with column filter, each thread computes just one pixel from reading N(filter length) neighbor pixels of buffer by sharing memory between threads of the same block
10: (**GPU memory transfer**) Store filtering results in Global memory
11: (**GPU→CPU memory transfer**) Copy filtering results to CPU memory

Algorithm 2 FFT2D Filtering CUDA Algorithm

1: (**load_image**) Load image to CPU memory
2: (**compute_filter**) Compute 2D filter on CPU
3: (**CPU→GPU memory transfer**) Copy image data and filter data to GPU Global memory
4: (**compute_FFT2D**) Convert image and filter to (Frequency-Domain) using "cuFFT" FFT2D
5: (**multiply"convolve"**) Perform the point-wise multiplication of the FFT of image and filter (Complex Number Multiplication), each thread processes just one pixel, reading one entry from image and one entry from filter to perform the two complex number multiplication
6: (**compute_IFFT2D**) Convert result to (Time-Domain) using "cuFFT" IFFT2D
7: (**GPU→CPU memory transfer**) Copy filtering result data to CPU memory

4. APPLICATION TO MEDICAL IMAGES

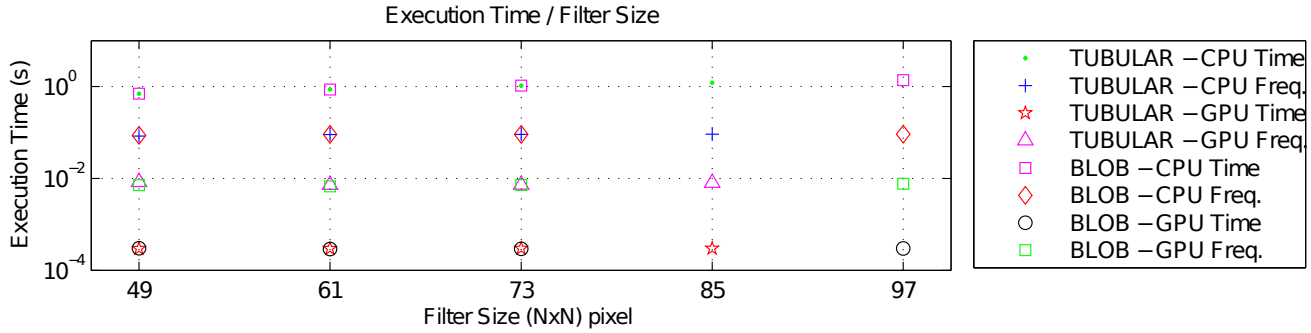
In this section we apply the methodoloy proposed in this work to wireless capsule endoscopy (WCE) images with 576×576 pixel and correspondent resized images with 1728×1728 pixel (in order to demonstrate the potential of this approach for high resolution images).

A sequential version of blob and tubular detectors was utilized in [1,2] for developing an automated algorithm to distinguish between abnormal (bleeding and/or blood) and normal images. We refer the reader to [1] for more details.

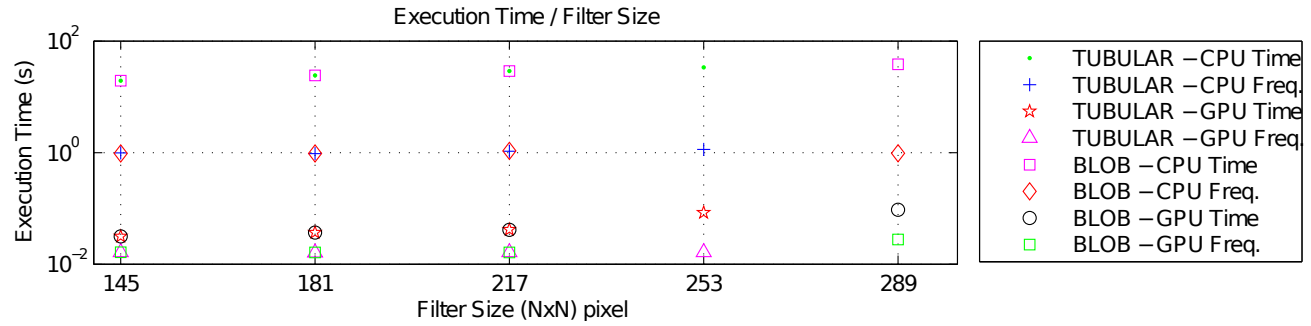
The program is developed using CUDA driver 5.5 and the C/C++ code compiled with GCC-4.6.3. The host system has an Intel Core i7 950 CPU @ 3.07GHz and runs the GNU/Linux kernel 3.8.0-31-generic. The GPU device consists of a Geforce GTX 680 with 1536 CUDA cores.

The global filtering times, corresponding to the proposed the blob and tubular detectors, for 576×576 WCE images are shown in Fig. 2a), and for the resized 1728×1728 images are shown in Fig. 2b). The Fig. 3 (a) displays two examples of WCE images, having abnormalities, and columnn (b) shows the correspondent scalar input images for the blob and tubular detectors [1, 2]. The last column (c) exhibits the abnormal regions successfully detected (bleeding, with the shape of a blob, for the top image and blood, with the shape of a tube, for the bottom image).

As depicted comparing Fig. 2a) and Fig. 2b), the time-domain filtering on GPU performs faster than the frequency-domain for 576×576 images and the frequency-domain filtering on GPU performs faster for 1728×1728 images. The size of the filter applied depends on the size of the objects that we want to identify, consequently, larger images require larger filters, assuming that images is of the same scene. For smaller filters, the time-domain method is faster because this method uses shared memory and the shared memory is fast and seen by all threads within the same block. So we can have several threads processing the same local data to opti-



(a) Execution time for images with 576×576 pixel.



(b) Execution time for images with 1728×1728 pixel.

Fig. 2: Global filtering processing times for BLOB and TUBULAR shape detections, varying the filter size and platform. The tests were performed on WCE images, applying 3 filters for each dimension.

mize memory bandwidth, but when filter size increases the shared memory used increases too. Shared memory are typically small in size. Therefore, we need to reduce the block size and this action increases the amount of data exchanges with global memory and the number of memory accesses will increase and slow down the process. On the other hand, in the frequency-domain approach, for larger filters we can set a fixed block size, thus using global memory more efficiently.

5. CONCLUSIONS

We have devised a parallel signal processing framework for detecting blob and tubular structures in images which can be helpful in many computer vision applications. The proposed framework is applied to wireless capsule endoscopy images for detecting bleeding/blood regions. The filtering process represents the functionality with higher impact in the global processing time, so we implemented several versions and we conclude that time-domain approaches executing on GPU are faster for small filters and that frequency-domain GPU methods are more efficient for larger filters. Through parallelization of the algorithm, we obtain a speedup up to 17x on images with 576×576 pixel and up to 49x on images with 1728×1728 pixel. To the best of our knowledge, this is the first GPU-accelerated algorithm to process WCE images in order to speed up the findings of blood/bleeding regions. Furthermore, this proposed approach has the potential to be used in many other applications, as those mentioned in Section 1.

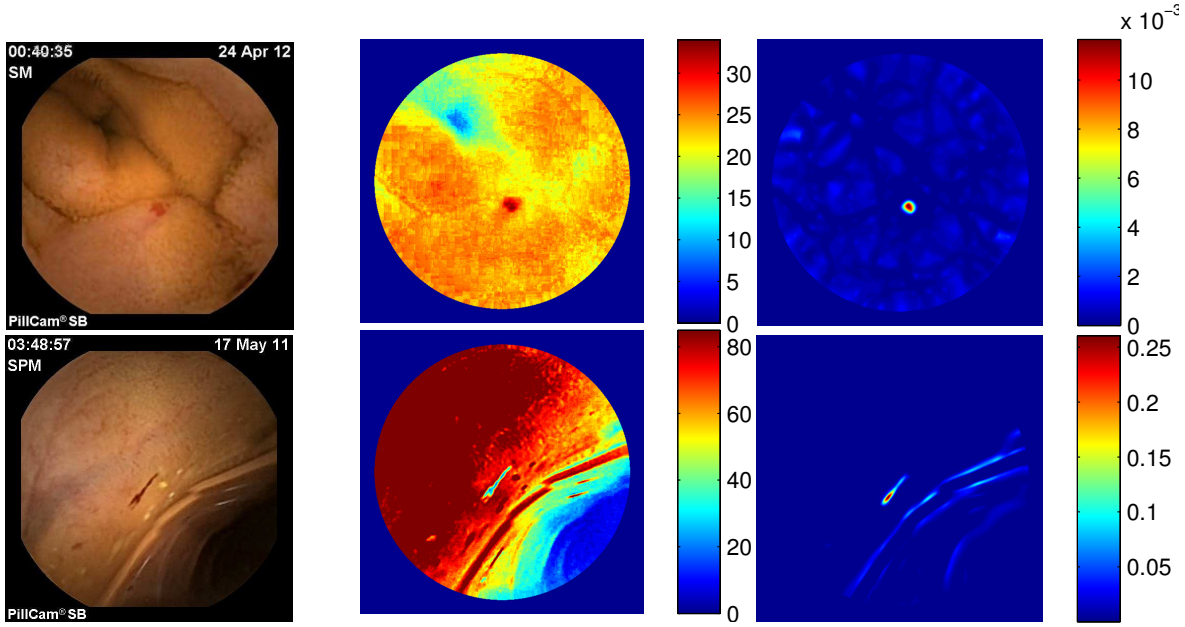
ACKNOWLEDGEMENTS

This work was partially supported by project PTDC/MATNA N/0593/2012 from CMUC and FCT (Portugal) through the European program COMPETE/ FEDER, and also by projects PEst-C/MAT/UI0324/2011 and PEst-OE/EEI/LA0008/2013 from Instituto de Telecomunicações.

REFERENCES

[1] I. N. Figueiredo, S. Kumar, Carlos Leal, and Pedro N. Figueiredo, "Computer-assisted bleeding detection in wireless capsule endoscopy images," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 1, pp. 198–210, 2013.

[2] I. N. Figueiredo, Sunil Kumar, Carlos Leal, and Pedro N. Figueiredo, "An automatic blood detection algorithm for wireless capsule endoscopy images," in *Computational Vision and Medical Image Processing,VIPIMAGE* 2013, *João Tavares & Natal Jorge (eds),* 2014 *Taylor & Francis Group, London, ISBN 978-1-138-00081-0 (ECCOMAS Thematic Conference*



(a) Original WCE images. (b) Pre-processed input images (c) Output detected bleeding/blood zones.

Fig. 3: From left to right: original input, intermediate images and detected regions of interest.

on Computational Vision and Medical Image Processing), pp. 237–241.

[3] G. Li, T. Liu, J. Nie, L. Guo, J. Malicki, A. Mara, S. A. Holley, W. Xia, , and S.T. Wong, "Detection of blob objects in microscopic zebrafish images based on gradient vector diffusion," *Cytometry A*, vol. 71, pp. 835–845, 2007.

[4] R. Manniesing, M. A. Viergever, and W. J. Niessen, "Vessel enhancing diffusion: A scale space representation of vessel structures," *Medical Image Analysis*, vol. 10, no. 6, pp. 815 – 825, 2006.

[5] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Multiscale vessel enhancement filtering," in *Medical Image Computing and Computer-Assisted Intervention*, Cambridge, MA, USA, 1998, pp. 130–137.

[6] G. Gerig, G. Szekely, G. Israel, and M. Berger, "Detection and characterization of unsharp blobs by curve evolution," in *In Proc. of Information Processing in Medical Imaging*, 165-176, 1995.

[7] S.A. Mahmoudi, F. Lecron, P. Manneback, M. Benjelloun, and S. Mahmoudi, "GPU-based segmentation of cervical vertebra in X-Ray images," in *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, 2010, pp. 1–8.

[8] M. Martins, G. Falcao, and I. N. Figueiredo, "Fast Aberrant Crypt Foci Segmentation on the GPU," International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, May, 2013.

[9] A. M. R. Schilham, B. van Ginneken, and M. Loogr, "Multi-scale nodule detection in chest radiographs," in *Lecture Notes in Computer Science, vol. 2878*, 2003.

[10] H. Lee, M. Harris, E. Young, and V. Podlozhnyuk, "Image convolution with CUDA," NVIDIA Corporation, 2007.

[11] Matteo Frigo and Steven G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, Special issue on "Program Generation, Optimization, and Platform Adaptation".

[12] V. Podlozhnyuk, M. Harris, and E. Young, "NVIDIA CUDA C programming guide," NVIDIA Corporation, 2012.

[13] V. Podlozhnyuk, "FFT-based 2D Convolution," NVIDIA Corporation, 2012.

B

Appendix B

A GPU accelerated algorithm for blood detection in wireless capsule endoscopy images

Sunil Kumar [1], Isabel N. Figueiredo [1], Carlos Graca [2] and Gabriel Falcao [2]

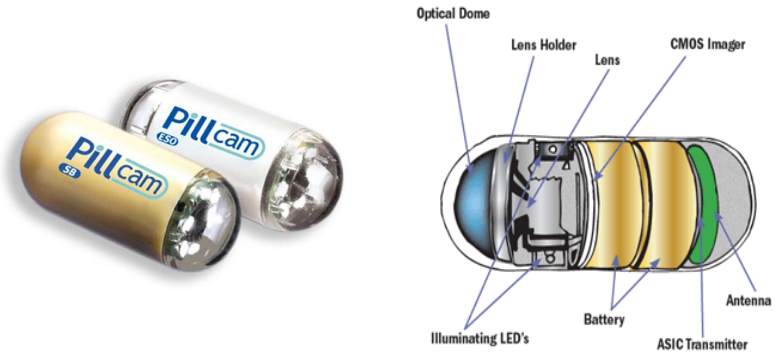
Abstract Wireless capsule endoscopy (WCE) has emerged as a powerful tool in the diagnosis of small intestine diseases. One of the main limiting factor is that it produces a huge number of images, whose analysis, to be done by a doctor, is an extremely time consuming process. Recently, we proposed [8] a computer-aided diagnosis system for blood detection in WCE images. While the algorithm in [8] is very promising in classifying the WCE images, it still does not serve the purpose of doing the analysis within a very less stipulated amount of time; however, the algorithm can indeed profit from a parallelized implementation. In the algorithm we identified two crucial steps, segmentation (for discarding non-informative regions in the image that can interfere with the blood detection) and the construction of an appropriate blood detector function, as being responsible for taking most of the global processing time. In this work, a suitable GPU-based (graphics processing unit) framework is proposed for speeding up the segmentation and blood detection execution times. Experiments show that the accelerated procedure is on average 50 times faster than the original one, and is able of processing 72 frames per second.

1 Introduction

Wireless capsule endoscopy (WCE), also called capsule endoscopy (CE), is a noninvasive endoscopic procedure which allows visualization of the small intestine, without sedation or anesthesia, which is difficult to reach by conventional endoscopies. As the name implies, capsule endoscopy makes use of a swallowable capsule that

[1] CMUC, Department of Mathematics, Faculty of Science and Technology, University of Coimbra, Portugal.

[2] Instituto de Telecomunicações, Department of Electrical and Computer Engineering, Faculty of Science and Technology, University of Coimbra, Portugal.



(a) (b)

Fig. 1 (a) Image of the capsule. (b) Interior of the capsule.

contains a miniature video camera, a light source, batteries, and a radio transmitter (see Figure 1). This takes continual images during its passage down the small intestine. The images are transmitted to a recorder that is worn on a belt around the patients waist. The whole procedure lasts 8 hours, after which the data recorder is removed and the images are stored on a computer so that physicians can review them and analyze the potential source of diseases. Capsule endoscopy is useful for detecting small intestine bleeding, polyps, inflammatory bowel disease (Crohn's disease), ulcers, and tumors. It was first invented by Given Imaging in 2000 [12]. Since its approval by the FDA (U.S. Food and Drug Administration) in 2001, it has been widely used in hospitals.

Although capsule endoscopy demonstrates a great advantage over conventional examination procedures, some improvements remain to be done. One major issue with this new technology is that it generates approximately 56,000 images per examination for one patient, which is very time consuming to analyse. Furthermore, some abnormalities may be missed because of their size or distribution, due to visual fatigue. So, it is of great importance to design a real-time computerized method for the inspection of capsule endoscopic images. *Given Imaging Ltd.* has also developed the so called RAPID software for detecting abnormalities in CE images. But its sensitivity and specificity, respectively, were reported to be only 21.5% and 41.8% [10], see also [19]. Recent years have witnessed some development on automatic inspection of CE images, see [1, 4, 5, 6, 14, 18, 20, 15, 9, 7].

The main indication for capsule endoscopy is obscure digestive bleeding [5, 14, 18, 20, 9]. In fact, in most of these cases, the source of the bleeding is located in the small bowel. However, often, these bleeding regions are not imaged by the capsule endoscopy. This is why the blood detection is so important when we are dealing with capsule endoscopy. The current work is an extension of the paper [8], where an automatic blood detection algorithm for CE images was proposed. Utilizing Ohta color channel (R+G+B)/3 (where R, G and B denote the red, green and blue chan-

nel, respectively, of the input image), we employed analysis of eigenvalues of the image Hessian matrix and multiscale image analysis approach for designing a function to discriminate between blood and normal frames. The experiments show that the algorithm is very promising in distinguishing between blood and normal frames. But, the algorithm is not able to process huge number of images produced by WCE examination of a patient, within a very less stipulated amount of time. However, the computations of the algorithm can indeed be parallelized, and thus, can process the huge number of images within a very less stipulated amount of time. In the algorithm we identified two crucial steps, segmentation (for discarding non-informative regions in the image that can interfere with the blood detection) and the construction of an appropriate blood detector function, as being responsible for taking most of the global processing time. We propose a suitable GPU-based framework for speeding up the segmentation and blood detection execution times, and hence the global processing time. Experiments show that the accelerated procedure is on average 50 times faster than the original one, and is able of processing 72 frames per second.

This chapter is structured as follows. A choice of the suitable color channel is made in Section 2.1 and segmentation of informative regions is done in Section 2.2. A blood detector function is introduced in Section 2.3. The outline of the the algorithm is given in Section 2.4. Validation of the algorithm on our current data set is provided in Section 3. The GPU procedure for speeding up the segmentation and blood detection is described in Section 4. Finally, the chapter ends with some conclusions in Section 5.

2 Blood detection algorithm

Notation: Let Ω be an open subset of R^2, representing the image (or pixel) domain. For any scalar, smooth enough, function u defined on Ω, $\|u\|_{L^1(\Omega)}$ and $\|u\|_{L^\infty(\Omega)}$, respectively, denote the L^1 and L^∞ norms of u.

2.1 Color space selection

Color of an image carries much more information than the gray levels. In many computer vision applications, the additional information provided by color can aid image analysis. The Ohta color space [17] is a linear transformation of the RGB color space. Its color channels are defined by $A_1 = (R+G+B)/3$, $A_2 = R-B$, and $A_3 = (2G-R-B)/2$. We observe that channel A_1 has the tendency of localizing quite well the blood regions, as is demonstrated in Figure 3. The first row corresponds to the original WCE images with blood regions and the second row exhibits their respective A_1 channel images. We also observe that, before computing the A_1 channel of the images, we applied an automatic illumination correction scheme [22] to the original images, to reduce the effect of illumination.

2.2 Segmentation

Many WCE images contain uninformative regions such as bubbles, trash, dark regions and so on, which can interfere with the detection of blood. More information on uninformative regions can be found in [1]. We observe that the second component (which we call henceforth a-channel) of the CIE Lab color space has the tendency of separating these regions from the informative ones. More precisely, for better removal of the uninformative regions, we first decompose the a-channel into geometric and texture parts using the model described in [2, Section 2.3], and perform the two phase segmentation. This latter relies on a reformulation of the Chan and Vese variational model [2, 3], over the geometric part of the a-channel.

The segmentation method is described as follows: We first compute the constants c_1 and c_2 (representing the averages of I in a two-region image partition). We then solve the following minimization problem

$$\min_{u,v}\left\{TV_g(u)+\frac{1}{2\theta}\|u-v\|^2_{L^2(\Omega)}+\int_\Omega\Big(\lambda\, r(I,c_1,c_2)\,v+\alpha\,\nu(v)\Big)\,dxdy\right\} \quad (1)$$

where $TV_g(u) := \int_\Omega g(x,y)|\nabla u|\,dxdy$ is the total variation norm of the function u, weighted by a positive function g; $r(I,c_1,c_2)(x,y) := \big(c_1-I(x,y)\big)^2-\big(c_2-I(x,y)\big)^2$ is the fitting term, $\theta>0$ is a fixed small parameter, $\lambda>0$ is a constant parameter weighting the fitting term, and $\alpha\,\nu(v)$ is a term resulting from a reformulation of the model as a convex unconstrained minimization problem (see [2, Theorem 3]). Here, u represents the two-phase segmentation and v is an auxiliary unknown. The segmentation curve, which divides the image into two disjoint parts, is a level set of u, $\{(x,y)\in\Omega:\ u(x,y)=\mu\}$, where in general $\mu=0.5$ (but μ can be any number between 0 and 1, without changing the segmentation result, because u is very close to a binary function).

The above minimization problem is solved by minimizing u and v separately, and iterated until convergence. In short we consider the following two steps:
1. v being fixed, we look for u that solves

$$\min_u\left\{TV_g(u)+\frac{1}{2\theta}\|u-v\|^2_{L^2(\Omega)}\right\}. \quad (2)$$

2. u being fixed, we look for v that solves

$$\min_v\left\{\frac{1}{2\theta}\|u-v\|^2_{L^2(\Omega)}+\int_\Omega\Big(\lambda\, r(I,c_1,c_2)\,v+\alpha\,\nu(v)\Big)\,dxdy\right\}. \quad (3)$$

It is shown that the solution of (2) is ([2, Proposition 3])

$$u=v-\theta\,\mathrm{div}\,p,$$

where div represents the divergent operator, and $p=(p_1,p_2)$ solves

$$g\nabla(\theta\,\mathrm{div}\,p-v)-|\nabla(\theta\,\mathrm{div}\,p-v)|p=0.$$

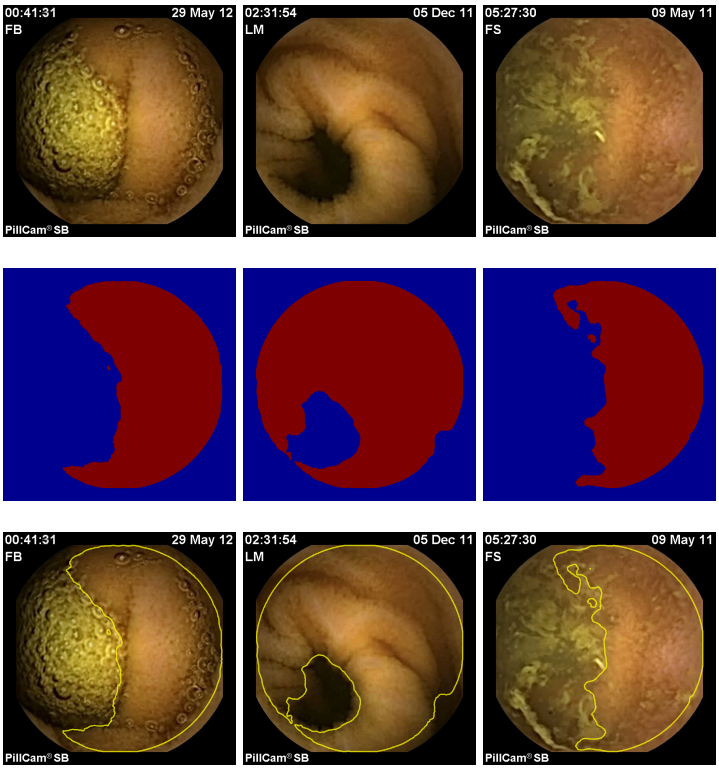


Fig. 2 First row: Original image. Second row: Segmentation mask. Third row: Original image with segmentation curve superimposed.

The problem for p can be solved using the following fixed point method

$$p^0 = 0,\ p^{n+1} = \frac{p^n + \delta t \nabla(\mathrm{div} p^n - v/\theta)}{1 + \frac{\delta t}{g}|\nabla(\mathrm{div} p^n - v/\theta)|}.$$

Again from [2, Proposition 4], we have

$$v = \min\{\max\{u - \theta\lambda r(I, c_1, c_2), 0\}, 1\}.$$

The segmentation results for some of the WCE images are shown in Figure 2. The first row corresponds to the original images, the second row shows the segmentation masks, and the third row displays the segmentation curves superimposed on the original images.

In these experiments (and also in the tests performed in section 3) the values chosen for the parameters involved in the definition of (1), are those used in [2], with g the following edge indicator function $g(\nabla u) = \frac{1}{1+\beta\|\nabla u\|^2}$ and $\beta = 10^{-3}$.

2.3 Detector function

We now introduce the detector function that is designed to discriminate between blood and non-blood frames. We resort to the analysis of eigenvalues of the image Hessian matrix and multiscale image analysis approach. Based on the eigenvalues, both blob-like and tubular-like structures can be detected. For a scalar image $I : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}$, we define the Hessian matrix of one point (x,y), and at a scale s, by

$$H_s(x,y) = \begin{pmatrix} I^s_{xx} & I^s_{xy} \\ I^s_{xy} & I^s_{yy} \end{pmatrix},$$

where I^s_{xx}, I^s_{xy} and I^s_{yy} are the second-order partial derivatives of I and the scale s is involved in the calculation of these derivatives. The Hessian matrix describes the second order local image intensity variations around the selected point. Suppose $\lambda_{s,1}$ and $\lambda_{s,2}$ are two eigenvalues of the Hessian matrix H_s. Further, suppose that $|\lambda_{s,1}| \leq |\lambda_{s,2}|$. Setting $F_s = \lambda^2_{s,1} + \lambda^2_{s,2}$, we define

$$F(x,y) = \max_{s_{min} \leq s \leq s_{max}} F_s(x,y), \tag{4}$$

where s_{min} and s_{max} are the minimum and maximum scales at which the blood regions are expected to be found. We remark that they can be chosen so that they cover the whole range of blood regions.

Setting now

$$f_1 = \exp\left(-\beta F_s^2\right) \quad \text{and} \quad f_2 = \left(1 - \exp\left(-\alpha \left(\frac{\lambda_{s,1}}{\lambda_{s,2}}\right)^2\right)\right),$$

and motivated from [11], we define the blob (B_s) and ridge (R_s) detectors (at each point of the domain)

$$B_s = \begin{cases} 0, & \text{if} \quad \lambda_{s,1}\lambda_{s,2} < 0 \quad \text{or} \quad |\lambda_{s,2} - \lambda_{s,1}| > \delta \\ (1-f_1)f_2, & \text{otherwise,} \end{cases} \tag{5}$$

and

$$R_s = \begin{cases} 0, & \text{if} \quad \lambda_{s,2} > 0, \\ (1-f_1)(1-f_2), & \text{otherwise.} \end{cases} \tag{6}$$

Here α and β are the parameters which control the sensitivity of the functions and δ is an user chosen threshold. We then compute the maximum for each scale

$$B(x,y) = \max_{s_{min} \leq s \leq s_{max}} B_s(x,y) \quad \text{and} \quad R(x,y) = \max_{s_{min} \leq s \leq s_{max}} R_s(x,y),$$

In the computations, we take $s = 8, 10, 12, 14$. The results of the functions F and the sum $B + R$, for blood and non-blood images are displayed in Figures 3 and 4, respectively.

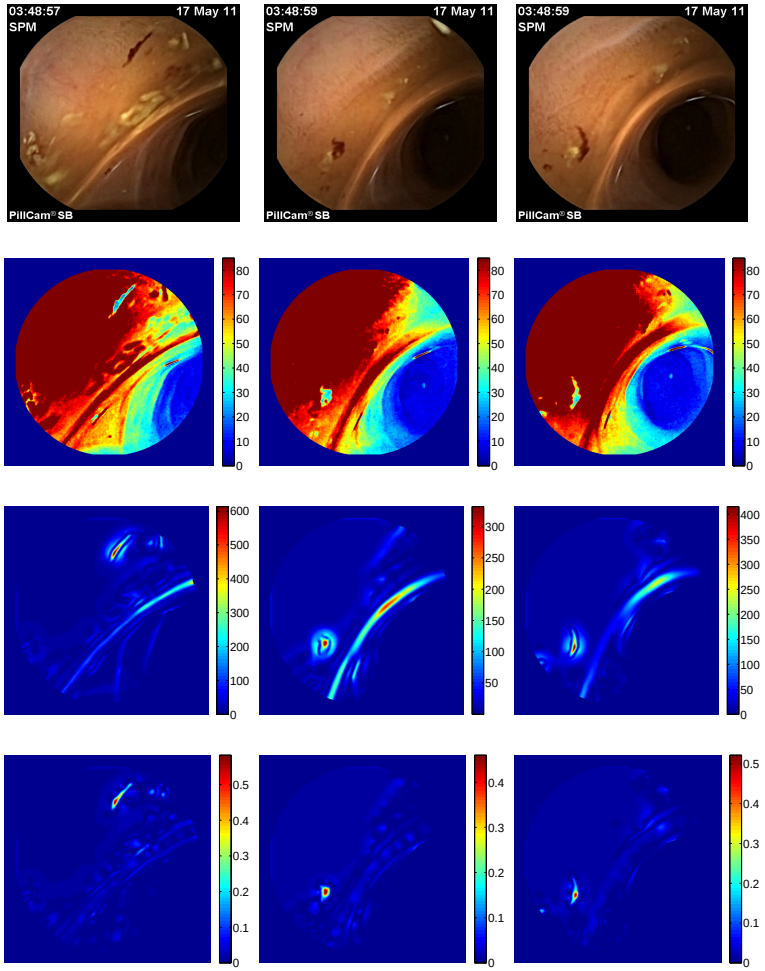


Fig. 3 First row: Original image with blood region. Second row: A_1 color channel. Third row: Function F. Fourth row: Function $B+R$.

We denote by $\widetilde{\Omega}$, in the image domain, the segmented region of I, that is, $\widetilde{\Omega} = \Omega \cap \Omega_{seg}$, where Ω_{seg} is the segmented sub-domain of I containing the blood. We use the intensity and gradient information of the above functions for designing our detector function, DF, which is defined by

$$DF = \frac{||F||_{L^\infty(\widetilde{\Omega})} ||B+R||_{L^\infty(\widetilde{\Omega})}}{||B+R||_{L^1(\widetilde{\Omega})}}.$$

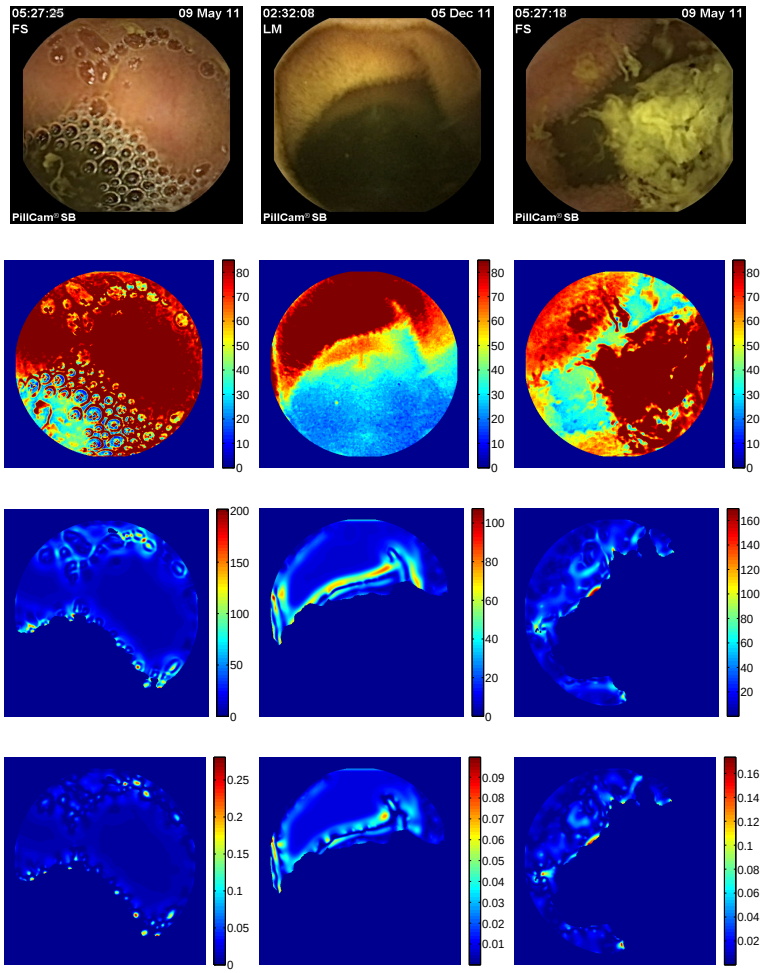


Fig. 4 First row: Original image without blood region. Second row: A_1 color channel. Third row: Function F. Fourth row: Function $B+R$.

2.4 Algorithm outline

For each WCE image the algorithm consists of the following four steps:

1. Firstly, we remove additional details (such as patient name, date and time) from the original image. For this purpose, we clip around the circular view of the original image. Next, we apply an automatic illumination correction scheme [22], for reducing the effect of illumination.
2. We then consider the Ohta color channel $(R+G+B)/3$ for the illumination corrected image.

3. We next apply the two-phase segmentation method [2] for removing uninformative regions (such as bubbles, trash, liquid, and so on) over the geometric part of the second component of the CIE Lab color space.
4. Finally, we compute the functions F, $B+R$ and the blood detector function DF.

3 Validation of the algorithm

We test the performance of the algorithm on a data set prepared by medical the experts. *Given Imaging*'s Pillcam SB capsule was used to collect the videos in the University Hospital of Coimbra. To make the data set representative, the images were collected from 4 patients video segments. The data set consists of 27 blood images and 663 normal images. We use standard performance measures: sensitivity, specificity and accuracy. These are defined as follows:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}},$$

$$\text{Accuracy} = \frac{\text{TN}+\text{TP}}{\text{TN}+\text{FP}+\text{TP}+\text{FN}},$$

where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives, respectively. For a particular decision threshold T, if for an image frame J, $DF > T$, it is a positive frame; if $DF \leq T$, it is a negative frame. If J belongs to the class of blood image frames and it is classified as negative, it is counted as a false negative; if it is classified as positive, it is counted as a true positive. If J belongs to the class of non-blood image frames and it is classified as positive, it is counted as a false positive; if it is classified as negative, it is counted as a true negative.

Sensitivity represents the ability of the algorithm to correctly classify an image as a frame containing blood, while specificity represents the ability of the algorithm to correctly classify an image as a non-blood frame. The third measure, accuracy, is used to assess the overall performance of the algorithm. There is also another performance measure commonly used in the literature, false alarm rate (FAR). However, it can be computed from the specificity: FAR=1-Specificity.

Receiver operating characteristic (ROC) curve is a fundamental tool for detection evaluation. In a ROC curve sensitivity is plotted in function of FAR. Each point on the ROC curve represents a sensitivity/FAR pair corresponding to a particular decision threshold. It shows the tradeoff between sensitivity and specificity. Figure 5 represents the ROC curve with respect to the function DF. For FAR$\leq$ 10%, the best sensitivity achieved is 70.37%. In particular, the sensitivity, FAR and accuracy obtained are 70.37%, 9.6% and 89.56%, respectively, for the threshold $2.8928E+007$. In summary, these results show that the presented algorithm is very promising for the detection of blood regions.

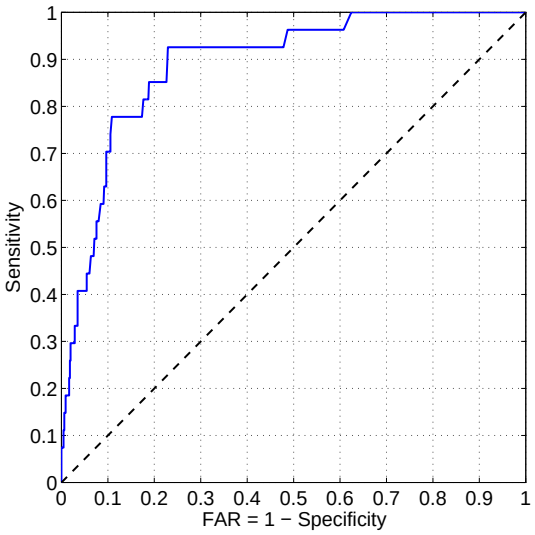


Fig. 5 ROC curve for function *DF*.

4 Speeding up the segmentation and detector performance

In this section we describe general facts about the apparatus specifications. In particular, we detail the GPUs adopted and the underlying architectures. Finally, we address the parallelization of the algorithms proposed, namely by detailing the segmentation and blood detector parallelization procedures on the GPU, and reporting the results obtained for the current medical dataset.

The pipeline of the algorithm, described in Section 2, has been first implemented on a CPU Intel Core i7 950 @ 3.07GHz, with 12GB of RAM, running a GNU/Linux kernel 3.8.0-31-generic. The C/C++ code was compiled using GCC-4.6.3.

In order to process more frames per second, the segmentation and blood detector steps have been paralellized, for executing on GPU NVidia C2050 and NVidia GTX 680, compiled using NVIDIA Compute Unified Device Architecture (CUDA) driver 5.5 [21].

4.1 General overview of the GPU architecture

The host system usually consists of a CPU that orchestrates the entire processing by sending data and launching parallel kernels on the GPU device. At the end of processing, it collects computed data from the device and terminates execution. The parallelization of segmentation and blood detection procedures is carried out using the CUDA parallel programming model, by exploiting the massive use of thread-

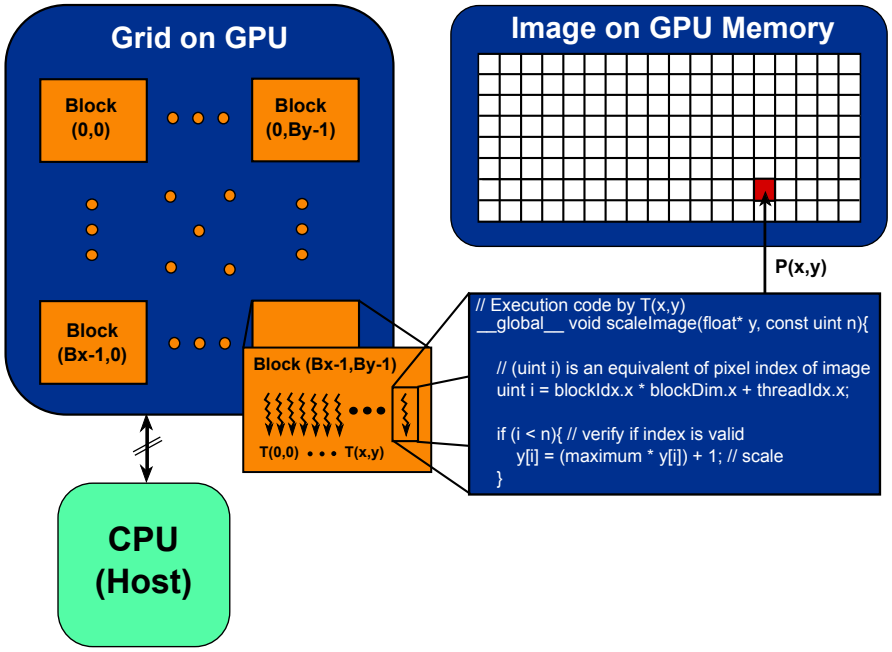


Fig. 6 Demonstration of the structure of a grid and thread blocks and how the same segment of code is executed by multiple threads. Each thread computes the result for one pixel.

and data-parallelism on the GPU. CUDA allows the programmer to write in a transparent way, scalable parallel C code [21] on GPUs.

As shown in Figure 6, each thread processes one pixel and thus multiple elements can be processed at the same time. This introduces a significant reduction in the global processing time of the proposed algorithm. When the host launches a parallel kernel, the GPU device executes a grid of thread blocks, where each block has a predefined number of threads executing the same code segment. Organized in groups of 32 threads (a warp), they execute synchronously and are time-sliced among the stream processors of each multiprocessor.

Figure 7 depicts a simplified overview of the GPU architecture. It shows that several multiprocessors contain a large number of stream processors (the number of stream processors and multiprocessors depends on the model and architecture of the GPU). In the present case, the NVidia GTX 680 GPU, which contains 8 multiprocessors with each multiprocessor containing 192 stream processors, performing a total of 1536 CUDA cores, executes the algorithm faster.

Before processing starts on the GPU, data is uploaded to device memory. This process is typically slow and consists in transferring the information from the host CPU memory to the GPU global memory (device). At the end of the processing, results are transferred from the GPU device global memory to the host CPU RAM memory.

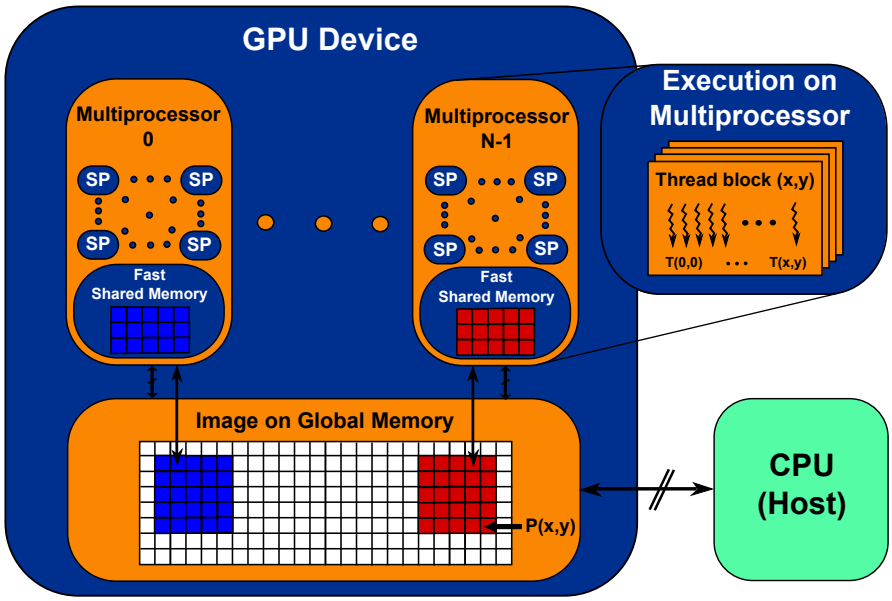


Fig. 7 Simplified GPU arquitecture. An example of how thread blocks are processed on GPU multiprocessors. A multiprocessor can execute more then one thread block concurrently.

In the GPU, there are several memory types and they have different impacts on the throughput performance. We highlight two of them:

- Global memory accesses are time consuming operations with high latency and may represent a bottleneck in the desired system's performance. Instead, coalesced accesses should be performed whenever possible. They imply data in global memory to be contiguously aligned, so that all 32 threads within a warp can access the respective 32 data elements concurrently on the same clock cycle, with thread T(x,y) accessing pixel P(x,y), as depicted in Figure 8.
- Also, modern GPUs have small and fast blocks of memory tightly coupled to the cores, which is shared by all threads within the same block. We can have several threads processing the same local data to optimize memory bandwidth (typically shared memory is faster than global memory when we need to share information among several threads), but shared memory is small in size. To maximize its use and performance, it is important to consider such size limitations. When large amounts of data have to be processed, data has to be partitioned in smaller blocks in order not to exceed the limits of shared memory. This action also represents penalties, since it increases the amount of data exchanges with global memory. Therefore, in the current work we use shared memory for calculating some procedures and global memory to perform the remaining functionalities, globally achieving an efficient memory usage as reported in later subsections.

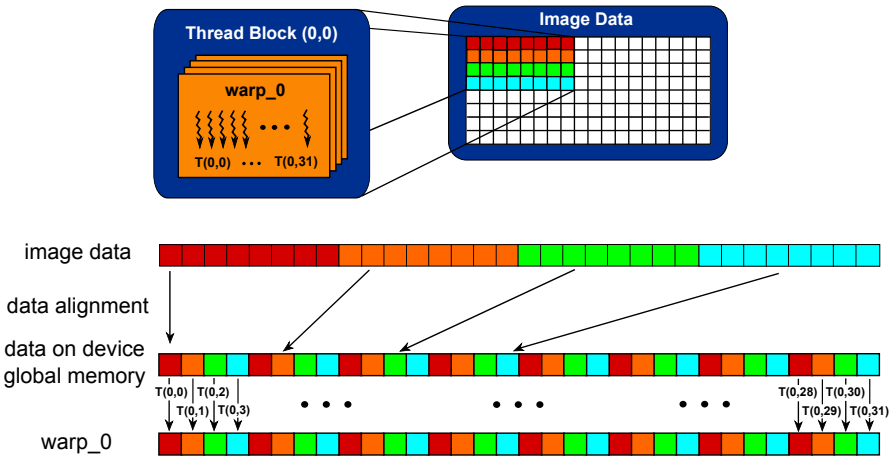


Fig. 8 Coalesced memory accesses illustrating a warp of 32 threads reading/writing the respective 32 data elements on a single clock cycle.

4.2 Segmentation parallelization

Some functions in the segmentation procedure, mentioned in Section 2.2, need to share image data between threads (e.g. neighboring pixels on the convolution procedure). Therefore, the use of shared memory is the best option to achieve a higher speedup (see [16] for a related work). These functions are: finding maximum and mean values, and 2D separable convolution [13]. All other functions perform slower if shared memory is used, because the total number of transactions to global memory will be greater.

The results of maximum and mean values are processed in two steps: the first step uses GPU grids with 256×256 block size; the second step uses 1×256 ; and in the 2D convolution, block sizes of dimension 16×16 are used.

The remaining functions in the segmentation step always use global memory and 1296×256 block sizes.

Processing Platform	Segmentation execution time (ms)	Segmentation (fps)
CPU Intel i7	240.0	4.2
GPU NVidia C2050	6.0	166.7
GPU NVidia GTX 680	4.8	208.3

Table 1 Computation times in milliseconds (ms) for the segmentation procedure and throughput measured in frames per second (fps). The tests were performed on WCE images with 576×576 pixels.

The computation times regarding the segmentation procedure are represented in Table 1, that shows the real speedups obtained using parallel computation on the

GPU; as displayed, this procedure runs 40 times faster on GPU NVidia C2050 and 50 times faster on GPU NVidia GTX 680, when compared to an Intel i7 CPU.

4.3 Blood detector parallelization

For speeding up the blood detector procedure, described in Section 2.3, we only use one function that shares image data between threads: 2D separable convolution [13]. The remaining functions perform slower if we use shared memory because the total number of transactions to global memory would assume a higher impact. The results of 2D separable convolution are computed using block sizes of dimension 16×16 and 8×8 for the scale values $s = [8\ 10]$ and $s = [12\ 14]$ (see Section 2.3), respectively. All other functions always use global memory blocks with size 8×8.

The computation times of the blood detector procedure are presented in Table 2. We clearly see the speedup obtained using parallel computation on GPU. This algorithm runs 58.9 times faster on GPU NVidia C2050 and 59.5 times faster on GPU NVidia GTX 680, when compared to an Intel i7 CPU.

Processing Platform	Blood Detector execution time (ms)	Blood Detector (fps)
CPU Intel i7	529.9	1.9
GPU NVidia C2050	9.0	111.1
GPU NVidia GTX 680	8.9	112.4

Table 2 Computation times in millisecons (ms) for the blood detector procedure and throughput measured in frames per second (fps). The tests were performed on WCE images with 576×576 pixels.

4.4 Speedup

Table 3 shows throughput measured in frames per second (fps) and the speedup of the full algorithm achieved. It can be seen that GPU NVidia GTX 680 is faster than NVidia C2050.

Processing Platform	Segmentation and Blood Detector (fps)	Speedup
CPU Intel i7	1.3	——-
GPU NVidia C2050	66.7	51.3 times faster
GPU NVidia GTX 680	72.9	56.1 times faster

Table 3 Throughput measured in fps and speedup archived to the complete algorithm (Segmentation and Blood Detector). Tests performed on WCE images with 576×576 pixels.

With the obtained speedup, the GPU NVidia GTX 680 shows to be able of processing 72 fps, which is equivalent to observe that the approximate total number of 56000 frames, generated by a complete WCE exam, can be computed in less than 13 minutes.

5 Conclusions

With the rapidly enhancing performances of graphics processors, improved programming support, and excellent price-to-performance ratio, GPUs have emerged as a competitive parallel computing platform for computationally expensive and demanding tasks in a wide range of medical image applications. We have proposed a GPU-based framework for blood detection in WCE images. The core of the algorithm lies in the definition of a good discriminator for blood and non-blood frames. This is accomplished by choosing a suitable color channel, image Hessian eigenvalue analysis and multiscale image analysis approach. Experimental results for our current dataset show that the proposed algorithm is effective, and achieves 89.56% accuracy. Moreover, it is shown that the accelerated procedure is on average 50 times faster than the original one, and is able of processing 72 frames per second. This is achieved by parallelizing the two crucial steps, segmentation and blood detector functionalities in the algorithm, that were consuming most of the global processing time. To perform these steps more efficiently we now run parallel code on GPUs with an appropriate use of memory (shared and global). This novel approach allows processing multiple pixels of an image at the same time, thus sustaining the obtained throughput levels.

Acknowledgements This work was partially supported by the project PTDC/MATNAN/0593/2012, and also by CMUC and FCT (Portugal), through European program COMPETE/ FEDER and project PEst-C/MAT/UI0324/2011. The work of Gabriel Falcao was also partially supported by Instituto de Telecomunicações and by the project PEst-OE/EEI/LA0008/2013.

References

1. Bashar, M., Kitasaka, T., Suenaga, Y., Mekada, Y., Mori, K.: Automatic detection of informative frames from wireless capsule endoscopy images. Medical Image Analysis **14**, 449–470 (2010)
2. Bresson, X., Esedoglu, S., Vandergheynst, P., Thiran, J.P., Osher, S.: Fast global minimization of the active contour/snake model. J Math. Imaging Vis. **28**, 151–167 (2007)
3. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. Image Processing **10**, 266–277 (2001)
4. Coimbra, M., Cunha, J.: MPEG-7 visual descriptors-contributions for automated feature extraction in capsule endoscopy. IEEE Transactions on Circuits and Systems for Video Technology **16**, 628–637 (2006)

5. Cui, L., Hu, C., Zou, Y., Meng, M.Q.H.: Bleeding detetction in wireless capsule endoscopy images by support vector classifier. In: Proceedings of the 2010 IEEE Conference on Information and Automation, pp. 1746–1751. Harbin, China (June 2010)
6. Cunha, J.P.S., Coimbra, M., Campos, P., Soares, J.M.: Automated topographic segmentation and transit time estimation in endoscopic capsule exams. IEEE Transactions on Medical Imaging **27**, 19–27 (2008)
7. Figueiredo, I.N., Kumar, S., Figueiredo, P.N.: An intelligent system for polyp detection in wireless capsule endoscopy images. In: Computational Vision and Medical Image Processing IV: VIPIMAGE 2013, ISBN: 9781315812922, pp. 229–235. Madeira Island, Funchal, Portugal (2013)
8. Figueiredo, I.N., Kumar, S., Leal, C., Figueiredo, P.N.: An automatic blood detection algorithm for wireless capsule endoscopy images. In: Computational Vision and Medical Image Processing IV: VIPIMAGE 2013, ISBN: 9781315812922, pp. 237–241. Madeira Island, Funchal, Portugal (2013)
9. Figueiredo, I.N., Kumar, S., Leal, C., Figueiredo, P.N.: Computer-assisted bleeding detection in wireless capsule endoscopy images. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization **1**, 198–210 (2013)
10. Francis, R.: Sensitivity and specificity of the red blood identification (RBIS) in video capsule endoscopy. In: 3rd INt. Conf. Capsule Endoscopy. Miami, FL, USA (Feb 2004)
11. Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M.A.: Multiscale vessel enhancement filtering. In: Medical Image Computing and Computer-Assisted Intervention, pp. 130–137. Cambridge, MA, USA (1998)
12. Idan, G., Meron, G., Glukhovsky, A.: Wireless capsule endoscopy. Nature **405**, 417–417 (2000)
13. Lee, H., Harris, M., Young, E., Podlozhnyuk, V.: Image convolution with CUDA. NVIDIA Corporation (2007)
14. Li, B., Q.-H-Meng, M.: Computer-aided detection of bleeding regions for capsule endoscopy images. IEEE Transactions on Biomedical Engineering **56**, 1032–1039 (2009)
15. Liedlgruber, M., Uhl, A.: Computer-aided decision support systems for endoscopy in the gastrointestinal tract: a review. IEEE Reviews in Biomedical Engineering **4**, 73–88 (2011)
16. Martins, M., Falcao, G., Figueiredo, I.N.: Fast aberrant crypt foci segmentation on the GPU. In: ICASSP'13: Proceedings of the 36th IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE (2013)
17. Ohta, Y.I., Kanade, T., Sakai, T.: Color information for region segmentation. Computer Graphics and Image Processing **13**, 222–241 (1980)
18. Pan, G., Xu, F., Chen, J.: A novel algorithm for color similarity measurement and the application for bleeding detection in WCE. I.J. Image, Graphics and Signal Processing **5**, 1–7 (2011)
19. Park, S.C., Chun, H.J., Kim, E.S., Keum, B., Seo, Y.S., Kim, Y.S., Jeen, Y.T., Lee, H.S., Um, S.H., Kim, C.D., Ryu, H.S.: Sensitivity of the suspected blood indicator: An experimental study. World J. Gastroenterolgy **18(31)**, 4169–4174 (2012)
20. Penna, B., Tilloy, T., Grangettoz, M., Magli, E., Olmo, G.: A technique for blood detection in wireless capsule endoscopy images. In: 17th European Signal Processing Conference (EUSIPCO 2009), pp. 1864–1868 (2009)
21. Podlozhnyuk, V., Harris, M., Young, E.: NVIDIA CUDA C programming guide. NVIDIA Corporation (2012)
22. Zheng, Y., Yu, J., Kang, S.B., Lin, S., Kambhamettu, C.: Single-image vignetting correction using radial gradient symmetry. In: Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08), pp. 1–8. Los Alamitos, Calif., USA (June 2008)

C

Appendix C

Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

Carlos Graca [1] · **Gabriel Falcao** [1] · **Isabel N. Figueiredo** [2] · **Sunil Kumar** [2]

Hybrid GPU-GPU computing: accelerated kernels for segmentation and object detection with medical image processing applications

Received: date / Revised: date

Abstract The last two decades have seen an amazing development of image processing techniques targeted for medical applications. We propose hybrid GPU-GPU based parallel algorithms for segmentation and object detection, aiming at accelerating two medical image processing methods: automated blood detection in wireless capsule endoscopy (WCE) images and automated bright lesion detection in retinal fundus images. In the former method we identified segmentation and object detection as being responsible for consuming most of the global processing time. While in the latter, as segmentation was not used, object detection was the compute-intensive task identified. Experimental results show that the accelerated method running on hybrid GPU-GPU systems for blood detection in WCE images is on average 168 times faster than the original CPU version and is able to process 218 frames per second. By applying the hybrid GPU-GPU framework for bright lesion detection in fundus images we are able to process 30 frames per second with a speedup average 324 times faster than the equivalent CPU version.

Keywords Segmentation, Shape-based object detection, Wireless capsule endoscopy, Fundus images, Automated diagnosis, Parallel image processing, Hybrid GPU-GPU Systems

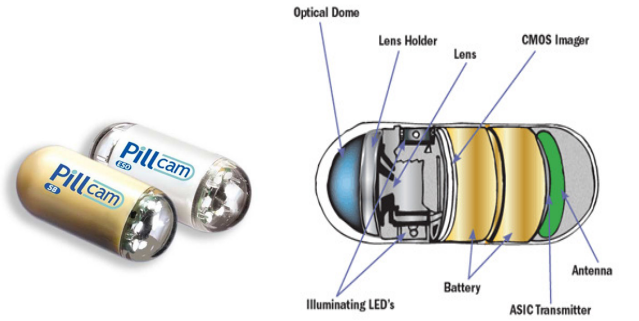
1 Introduction

Segmentation and object detection are two fundamental problems in computer vision which have been a major focus of research activities. Segmentation is the process of partitioning an image into distinct regions containing pixels with similar attributes. Many computer vision applications apply image segmentation techniques during preprocessing to reduce image information for increased processing efficiency. Some of the applications of image segmentation are in locating tumors and other pathologies, in satellite images (roads, forests, crops, etc.), in face recognition, in finger print recognition, in locating aberrant Crypt Foci, and others. In the problem for object detection we are interested in shape-based object detection, in particular objects that have blob/tubular shapes. Blob/tubular detection refers to methods that are aimed at detecting clustered points in the image that are either brighter or darker than the surrounding region. Detection of blob and/or tubular structures in images is an important step in the analysis of large-scale scientific data, as for example, detection of bleeding/blood regions in WCE images, bright lesions in fundus images, nodule detection in thorax x-ray images, nuclei detection in microscopic zebrafish images, enhancement of vascular structures, detection of lesions in images of multiple sclerosis patients, to name a few.

The purpose of this paper is to develop hybrid GPU-GPU based parallel algorithms for segmentation and shape-based object detection, aiming at accelerating two recently proposed medical image processing methods: automated blood detection in wireless capsule endoscopy (WCE) images [9] and automated bright lesions detection in retinal fundus images [7]. WCE (see Figure 1) has emerged as a powerful tool in the diagnosis of diseases of the gastrointestinal tract, and in particular of the small intestine. One of the main limiting factor of this technology is that it produces a huge number of images, whose analysis, to be performed by a doctor, is an extremely time consuming process. In recent years, we have seen some developments of the methods used for automatic inspection of WCE images, improving the detection of bleeding/blood, ulcers, polyps and tumors (see [2, 4, 5, 6, 16, 19, 20, 17, 10, 8, 14] and the references therein). All these methods were developed aiming at high classification accuracy. The natural next step would be to build a real-time method with high classification accuracy for automatically inspecting WCE images. With this goal in

[1] Instituto de Telecomunicações, Department of Electrical and Computer Engineering, Faculty of Science and Technology, University of Coimbra, 3030-290 Coimbra, Portugal.
E-mail: cgraca@co.it.pt E-mail: gff@co.it.pt

[2] CMUC, Department of Mathematics, University of Coimbra, 3001 - 501 Coimbra, Portugal.
E-mail: isabelf@mat.uc.pt E-mail: skumar@mat.uc.pt

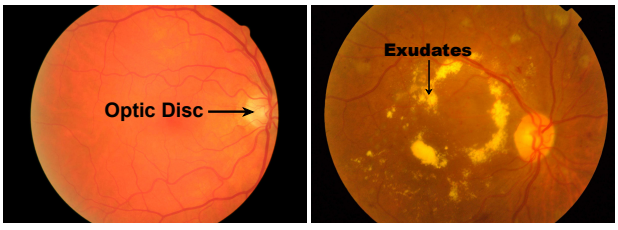


(a) (b)

Fig. 1 (a) Image of the PillCam SB capsule. (b) Interior and components of the capsule.

mind, we developed a real-time system for automated blood detection in WCE images. The current method consists of an accelerated version of the method in [9] that was shown high classification accuracy. In the method of [9] we identified two crucial steps, segmentation (for discarding non-informative regions in the image that can interfere with the blood detection) and shape-based object detection (for constructing an appropriate blood detector function) being responsible for consuming most of the global processing time.

Our second medical image processing method is related to diabetic retinopathy (DR), which is a serious sight threatening complication caused by diabetes. Early detection and treatment can limit the potential for significant vision loss caused from DR. Therefore, retinal fundus images (see Figure 2) are routinely ordered to a diabetic patient for any possible abnormality in eyes. DR lesions may be classified as red lesions, such as microaneurysms and hemorrhages, and bright lesions, such as exudates, drusen and cotton-wool spots. Automated detection and diagnosis of DR through the processing of retinal fundus images is a necessary step for the implementation of a large scale screening of diabetic patients. Recent years have witnessed developments of other methods, aiming at high classification accuracy, for detection of various DR lesions (see [7, 1, 22, 23] and the references therein). In this paper we develop an accelerated version of the method [7] proposed for automated detection of bright lesions in fundus images. In the algorithm shape-based object detection (for constructing an appropriate function for bright lesion detection) was the step responsible for consuming most of the global processing time. The retinal fundus images are captured using a Topcon TRC NW100 non-mydriatic retinal camera, thus producing large and compute-intensive high definition (HD) images.



(a) (b)

Fig. 2 a) A normal retina fundus image. b) Bright lesion visible in a retina fundus image.

As a first approach, segmentation and shape-based object detection algorithms using parallel C code are developed to run on a single-GPU (graphics processing units) multiprocessor. Next, a suitable hybrid GPU-GPU framework is proposed for speeding up the segmentation and shape-based detection execution times even further. In hybrid GPU-GPU assemblies we exploit a balanced distribution of GPU resources, thereby deciding how many images are processed on each GPU. This decision is made by a training process, which checks how many CUDA devices are available on the machine and runs several times the single-GPU versions in all available CUDA devices. With average of single-GPU execution times, we are able to calculate a suitable profile and workload distribution between all GPUs. This training process just needs to be performed once, in order to build a configuration file.

When we apply the segmentation and shape-based object detection on WCE images, experiments show that the accelerated single-GPU setup procedure is on average 92 times faster than the original one executed on CPU and is capable of processing 119 frames per second. Our best hybrid GPU-GPU approach is on average 168 times faster than the original one (CPU version) and is able to process 218 frames per second. On HD retinal fundus images we only have to apply shape-based object detection, and the fastest single-GPU system can process 16 frames per second with an average speedup of 179 times. In hybrid GPU-GPU mode we are able to process 30 frames per second, with an average speedup of 324 when compared to the original CPU version. Note that in both applications, we use original images provided by professional medical imaging equipments without manipulation or resizing.

This paper is structured as follows. In Section 2 we describe segmentation and shape-based object detection approaches and their application in medical image processing; in particular, we analyse the case studies of blood detection in WCE images and bright lesion detection in fundus images. Parallelization of the proposed methods is described in 3. Illustration of some results is provided in Section 4, while reported speedups using single-GPU assemblies are described in Section 5. A detailed time analysis of hybrid GPU-GPU computing is presented in Section 6. Finally, we close the paper in Section 7.

2 Segmentation and object detection approaches

2.1 Variational image segmentation

The segmentation method relies on a reformulation of the Chan and Vese variational model (see [3]), and is briefly de-

Temporarily unavailable for review

helpful in many computer vision applications. The proposed framework is applied to WCE images for detecting blood regions, and to HD retinal fundus images for detecting bright lesion regions. The present hybrid GPU-GPU approach is capable to process blood detection and bright lesion detection procedures 1.83 and 1.81 times faster, respectively, than with the best single-GPU version (NVidia GTX TITAN). This speedup is achieved by parallelizing two crucial steps, segmentation and shape-based object detection functionalities in the algorithm, that were consuming most of the global processing time. In order to perform these steps more efficiently, we make an appropriate use of memory (shared and global) supported by parallel C code running on GPUs. This novel approach allows processing multiple pixels of an image concurrently, and hybrid GPU-GPU assemblies allows to process more than one image at the same time in distinct devices, thus sustaining the obtained throughput levels.

It is shown that in the blood detection on WCE images the accelerated procedure running on faster single-GPU version is on average 92 times faster than the original sequential CPU version, and is able of processing 119 frames per second. The proposed hybrid GPU-GPU system with Dual GPU NVidia GTX TITAN shows to be capable of processing 218 fps, which allows that the approximate total number of 56000 frames, generated by a complete WCE exam, can be computed in less than 5 minutes. In HD retinal fundus images only shape-based object detection is used, and the fastest single-GPU system can process 16 frames per second with an average speedup of 179 times compared to sequential CPU version. In the proposed hybrid GPU-GPU system we can process 30 frames per second with an average speedup 324 times faster than the original CPU version. With such high throughputs we are able to build real-time systems to automatically detect bright lesions in fundus images and blood in WCE images, which may help the medical practitioner improving the diagnosis procedure.

Acknowledgements This work was partially supported by the project PTDC/MATNAN/0593/2012, and also by CMUC and FCT (Portugal), through European program COMPETE/FEDER and project PEst-C/MAT/UI0324/2011. The work was also partially supported by Instituto de Telecomunicações and by project PEst-OE/EEI/LA0008/2013.

References

1. Akram, M.U., Tariq, A., Khan, S.A., Javed, M.Y.: Automated detection of exudates and macula for grading of diabetic macular edema. Computer Methods and Programs in Biomedicine **114**, 141–152 (2014)
2. Bashar, M., Kitasaka, T., Suenaga, Y., Mekada, Y., Mori, K.: Automatic detection of informative frames from wireless capsule endoscopy images. Medical Image Analysis **14**, 449–470 (2010)
3. Bresson, X., Esedoglu, S., Vandergheynst, P., Thiran, J.P., Osher, S.: Fast global minimization of the active contour/snake model. J Math. Imaging Vis. **28**, 151–167 (2007)
4. Coimbra, M., Cunha, J.: MPEG-7 visual descriptors-contributions for automated feature extraction in capsule endoscopy. IEEE Transactions on Circuits and Systems for Video Technology **16**, 628–637 (2006)
5. Cui, L., Hu, C., Zou, Y., Meng, M.Q.H.: Bleeding detetction in wireless capsule endoscopy images by support vector classifier. In: Proceedings of the 2010 IEEE Conference on Information and Automation, pp. 1746–1751. Harbin, China (2010)
6. Cunha, J.P.S., Coimbra, M., Campos, P., Soares, J.M.: Automated topographic segmentation and transit time estimation in endoscopic capsule exams. IEEE Transactions on Medical Imaging **27**, 19–27 (2008)
7. Figueiredo, I.N., Kumar, S.: Wavelet-based computer-aided detection of bright lesions in retinal fundus images. In: Y. Zhang, J. Tavares (eds.) Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications, *Lecture Notes in Computer Science*, vol. 8641, pp. 234–240 (2014)
8. Figueiredo, I.N., Kumar, S., Figueiredo, P.N.: An intelligent system for polyp detection in wireless capsule endoscopy images. In: Computational Vision and Medical Image Processing IV: VIPIMAGE 2013, ISBN: 9781315812922, pp. 229–235. Madeira Island, Funchal, Portugal (2013)
9. Figueiredo, I.N., Kumar, S., Leal, C., Figueiredo, P.N.: An automatic blood detection algorithm for wireless capsule endoscopy images. In: Computational Vision and Medical Image Processing IV: VIPIMAGE 2013, ISBN: 9781315812922, pp. 237–241. Madeira Island, Funchal, Portugal (2013)
10. Figueiredo, I.N., Kumar, S., Leal, C., Figueiredo, P.N.: Computer-assisted bleeding detection in wireless capsule endoscopy images. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization **1**, 198–210 (2013)
11. Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M.A.: Multiscale vessel enhancement filtering. In: Medical Image Computing and Computer-Assisted Intervention, pp. 130–137. Springer, Heidelberg (1998)
12. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation"
13. Graca, C., Falcao, G., Kumar, S., Figueiredo, I.N.: Cooperative use of parallel processing with time or frequency-domain filtering for shape recognition. In: EUSIPCO 2014 (22nd European Signal Processing Conference 2014) (EUSIPCO 2014). Lisbon, Portugal (2014)
14. Kumar, S., Figueiredo, I.N., Graca, C., Falcao, G.: A gpu accelerated algorithm for blood detection in wireless capsule endoscopy images. In: Tavares, J.M. and Renato, R.S.N.J. (eds) Developments in Medical Image Processing and Computational Vision. Lecture Notes in Computational Vision and Biomechanics. Springer (2014)

15. Lee, H., Harris, M., Young, E., Podlozhnyuk, V.: Image convolution with CUDA. NVIDIA Corporation (2007)
16. Li, B., Q.-H-Meng, M.: Computer-aided detection of bleeding regions for capsule endoscopy images. IEEE Transactions on Biomedical Engineering **56**, 1032–1039 (2009)
17. Liedlgruber, M., Uhl, A.: Computer-aided decision support systems for endoscopy in the gastrointestinal tract: a review. IEEE Reviews in Biomedical Engineering **4**, 73–88 (2011)
18. Martins, M., Falcao, G., Figueiredo, I.N.: Fast aberrant crypt foci segmentation on the GPU. In: ICASSP'13: Proceedings of the 36th IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE (2013)
19. Pan, G., Xu, F., Chen, J.: A novel algorithm for color similarity measurement and the application for bleeding detection in WCE. I.J. Image, Graphics and Signal Processing **5**, 1–7 (2011)
20. Penna, B., Tilloy, T., Grangettoz, M., Magli, E., Olmo, G.: A technique for blood detection in wireless capsule endoscopy images. In: 17th European Signal Processing Conference (EUSIPCO 2009), pp. 1864–1868 (2009)
21. Podlozhnyuk, V., Harris, M., Young, E.: NVIDIA CUDA C programming guide. NVIDIA Corporation (2012)
22. Usman Akram, M., Khalid, S., Tariq, A., Khan, S.A., Azam, F.: Detection and classification of retinal lesions for grading of diabetic retinopathy. Computers in biology and medicine **45**, 161–71 (2014)
23. Zhang, X., Thibault, G., Decencire, E., Marcotegui, B., La, B., Danno, R., Cazuguel, G., Quellec, G., Lamard, M., Massin, P., Chabouis, A., Victor, Z., Erginay, A.: Exudate detection in color retinal images for mass screening of diabetic retinopathy. Medical Image Analysis **18**, 1026 – 1043 (2014)