FAST ABERRANT CRYPT FOCI SEGMENTATION ON THE GPU

Marco Martins[⋆] *Gabriel Falcao*[⋆] *Isabel N. Figueiredo*[†]

[⋆] Instituto de Telecomunicações, Department of Electrical and Computer Engineering
University of Coimbra, 3030-290 Coimbra, Portugal
[†]CMUC, Department of Mathematics
University of Coimbra, 3001-501 Coimbra, Portugal

ABSTRACT

Aberrant crypt foci (ACF) are thought to be one of the earliest manifestations of colorectal cancer (CRC). We propose a parallel signal processing algorithm (based on a fast approach of the active contour without edges model) to segment *in vivo* ACF, from stained endoscopic images, to serve as a diagnosis/prognosis auxiliary. In order to tackle this challenge, we successfully overcame several obstacles, involving the adaptation of the algorithm and data structures to a parallel environment. This algorithm was developed using the compute unified device architecture (CUDA) interface, which allows programming the graphics processing units (GPU). We show that it is possible to process four frames of 1 Mega pixel images per second using a conventional GPU. The GPU speeds up the *in vivo* ACF segmentation more than $40\times$, regarding to CPUs, and paves the way for real-time segmentation.

Index Terms— Aberrant Crypt Foci, Segmentation, CUDA, GPU, Colorectal Cancer

1. INTRODUCTION

Colorectal cancer (CRC) is one of the most frequent types of malignant tumours. In the United States it is the third most common fatal malignancy [1], and in Europe it is the second most frequent malignant disease [2]. Nevertheless, the prevention of CRC can be done more efficiently since its progression takes over 10-20 years, allowing sufficient time to detect the disease before it poses a clinical threat [3].

Aberrant crypt foci (ACF) are considered an earlier manifestation of CRC [4, 5]. ACF can be identified *in vivo* with magnification chromoscopic endoscopy [6]. Present methods used for evaluating ACF patterns by doctors, are rather subjective and not standardized. They rely on direct medical observation, due to the absence of computerized methods. Therefore, for doctors, a reliable computerized and fast method would be very helpful for evaluating ACF's pattern images, allowing a quicker and more accurate diagnosis, which could eventually be used to standardize the evaluation. This can be achieved by developing new systems based on image processing methods [7], that ideally would work in real-time.

Although parallel computing on GPU is a relatively new technology, it can surpass the central processing unit (CPU) computational power by many times and is used in a vast set of compute intensive applications that range from inference calculation [8] to contour detection [9, 10, 11, 12, 13, 14]. In this paper, we propose a parallel algorithm that suits fast ACF segmentation on GPUs. We show how the particularities of this signal processing algorithm can be accommodated in order to exploit multi-thread based execution on many-core GPUs [15], avoiding the use of divergent instructions, minimizing data transfers between different levels of the system's memory hierarchy, and coalescing data accesses. We conclude that an algorithm which outputs several frames per second is achievable on conventional GPUs. This can contribute to an earlier diagnosis of a large group of the population, and thus has the potential to apply signal processing methods to enhance the performance of human healthcare systems.

2. BACKGROUND, MOTIVATION AND SEGMENTATION MODEL

In 1987 Bird identified the first ACF [16], and since then they have been a case of study. Bird also hypothesized that they were possible precursors of CRC [5], and future studies supported that hypothesis [17]. When observed with stereomicroscopy and stained with methylene blue, the ACF are larger, thicker and darker staining than normal crypts [16], having noticeable oval, circular, and dilated or slit-like irregular openings [18].

Although we can find segmentation models proposed in the literature, a GPU implementation of any of these models would bring the processing of medical images to near real-time. In this paper we focus on a particular segmentation variational model for which we develop a new GPU-aware approach. The segmentation model is described in [19] and consists of an optimization of the active contour without edges model of Chan and Vese [20]. It corresponds to the following minimization problem:

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

where $r(x,c_1,c_2)=(c_1-(x))^2-(c_2-I(x))^2$ represents the fitting term, with I the given image, and c_1/c_2 the averages of I, inside/outside the segmentation curve (c_1 and c_2 are updated during the iterative procedure, starting with initial given guesses). The unknown u is the two-phase piecewise constant segmentation of I, the unknown v is an auxiliary function (forced to be equal to u), $TV_g(u)$ represents the total variation of u, weighted by an edge detection function g, $\theta>0$ is a fixed small parameter, $\lambda>0$ is a constant parameter, weighting the fitting term, $\alpha v(v)$ is a term resulting from a reformulation of the model, as a convex unconstrained minimization problem (see theorem 3, in [19]), and finally Ω is the image (or pixel) domain.

The problem in (1) can be computed by minimizing with respect to u and v separately, as follows:

• v being fixed, u is a solution of:

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

• u being fixed, v is a solution of:

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

Using a dual formulation of the TV (total variation) norm, the solution of (2) is defined by (see proposition 3, in [19]):

$$u=v-\theta\,\mathrm{div}\,p, \quad (4)$$

where div denotes the divergent operator and $p=(p^1,p^2)$ verifies

$$g(x)\,\nabla(\theta\,\mathrm{div}\,p-v)-|\nabla(\theta\,\mathrm{div}\,p-v)|\,p=0. \quad (5)$$

Moreover, the solution of (3) is (see proposition 4, in [19]):

$$v=\min\{\max\{u(x)-\theta\lambda r(x,c_1,c_2),0\},1\}. \quad (6)$$

Additionally, the solution p in (5) can be solved by a fixed point method:

$$p^0=0,\ p^{n+1}=\frac{p^n+\delta t\nabla\left(\mathrm{div}\,(p^n)-\frac{v}{\theta}\right)}{1+\frac{\delta t}{g(x)}\nabla\left(\mathrm{div}\,(p^n)-\frac{v}{\theta}\right)}, \quad (7)$$

where ∇ is the gradient operator.

Thus the algorithm leading to the solution of (2), consists in iterating (7) and (6) successively (the number of iterations is hereafter denoted by $iterno$). The constants c_1 and c_2 in the fitting term are updated periodically (a pre-defined number of iterations, hereafter called $updtFT$) using the formula:

$$r(x,c_1,c_2)=(\hat{u}_{in}-I(x))^2-(\hat{u}_{out}-I(x))^2, \quad (8)$$

where $\hat{u}_{in}$ and $\hat{u}_{out}$ are the averages of f, in the current segmented regions $\{x\in\Omega:\ u(x)\geq 0.5\}$ and $\{x\in\Omega:\ u(x)<0.5\}$, respectively, and $\{x\in\Omega:\ u(x)=0.5\}$ is the segmentation curve for the current iterate u.

3. PROPOSED PARALLEL ALGORITHM FOR GPU

The GPU is a highly parallel, multi-threaded, many-core processor [21] that largely surpasses CPUs in arithmetic throughput and memory bandwidth. Therefore, GPUs suit the speedup of a diversity of data-parallel signal processing applications [21], when properly orchestrated by a host system, typically a CPU. The compute unified device architecture (CUDA) is a programming interface that allows the programmer to write in a transparent way, scalable parallel C code [21] 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 piece of code (figure 1 depicts this hierarchy). Organized in groups of 32 threads (a warp), they execute synchronously and are time-sliced among the stream-processors of each multiprocessor.

3.1. Parallelization strategy of the algorithm

The algorithm described in section 2 was implemented using the CUDA toolkit to execute on the GPU, exploiting the massive use of thread- and data-parallelism.

Coalesced accesses and the use of shared memory: In a GPU, accesses to global memory are time consuming operations and may represent a bottleneck in the desired performance. Instead, coalesced accesses should be performed whenever necessary. They imply data in global memory to be contiguously aligned, so that all accesses in a warp are done concurrently, with thread T_k accessing pixel P_k. Also, we maximize the use of fast shared memory, which is tightly coupled to the processors. As depicted in figure 2, we store 512 data elements in shared memory for faster processing.

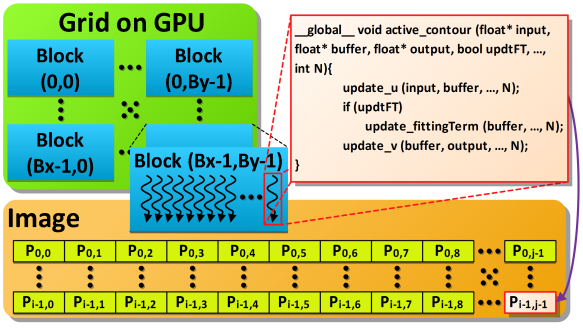


Fig. 1. The image, the grid of size (Bx,By) which computes the results for the image, the blocks of size (Tw,Tz) with threads, calculating the result for each pixel.

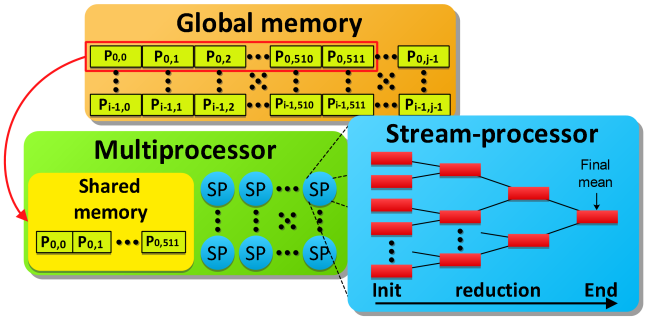


Fig. 2. Image in global memory and a representative subset of pixels stored in shared memory to perform the fast parallel reduction for mean calculations in (8). Each thread reduces a portion of data to be processed, ensuring data correctness.

Parallel active contour without edges detection model: The most intensive subkernel to process performs the update of u as indicated in (4). This operation involves the update of p^{n+1} regarding to iteration n as shown in (7) and is performed by each thread over a single pixel. Finite difference methods are used to calculate the mathematical divergence and gradient operators. We were able to exploit a high level of thread-based parallelism since each thread processes a different pixel, obtaining a global high throughput. The other two subkernels compute, respectively, the fitting term (8) and v as defined in (6). Altogether, these computations are iterated a predefined number of times to find the solution of the minimization problem in (1).

Avoiding divergent instructions: In order to maximize the full thread-level parallelism available on the GPU, the algorithm avoids whenever possible the usage of divergent instructions (for example by unrolling a loop), which otherwise serializes the algorithm, since threads follow different execution paths that cannot be executed simultaneously. However, the use of divergent instructions could not be completely avoided due to the border cases of the image, which penalizes parallel execution performance on the GPU due to the need of using conditional instructions.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

The program was developed using CUDA driver 5.0 with runtime 4.2 and the C/C++ code compiled with GCC-4.4.7. The host has an Intel Core i7 950 @ 3.07GHz and runs the GNU/Linux kernel 3.2.1-030201-i7. The device consists of a GPU Geforce GTX 680 with 1536 CUDA cores.

4.2. ACF segmentation on the Geforce GTX 680

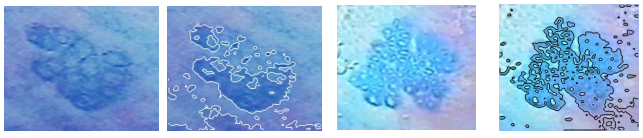
Using as input the grayscale images presented in figure 3 (a) and (c), the corresponding segmentation results are shown in figure 3 (b) and (d). Here, we can observe some false positives, nevertheless, the ACFs are segmented in an acceptable way, successfully identifying the crypt orifices, which is extremely relevant for the physicians (full size figures 3 and 4 are available at http://www.co.it.pt/acfgpu).

	Figure 3		Figure 4			
	(b)	(d)	D	L	C	R
λ	10^{-1}	10^{-1}	10^{-1}	10^{-4}	10^{-2}	10^{-1}
θ	1	1	1	10^{-1}	$1/2$	1
β	10^{-4}	10^{-4}	10^{-4}	1	10^{-2}	10^{-4}
δt	$1/8$	$1/8$	$1/8$	$1/32$	$1/16$	$1/8$
$iterno$	30	30	500	50	200	500
$updtFT$	30	10	5	100	30	5

Table 1. Parameters for the different images, with L, C and R denoting left, center and right, respectively, and D default value. The parameter λ is used to weight the fitting term, θ is a small constant forcing v to approach u, β is a weight in the edge detection function g, δt is the temporal step, $iterno$ is the number of iterations for the minimization process and $updtFT$ is the iterations period to update the fitting term.

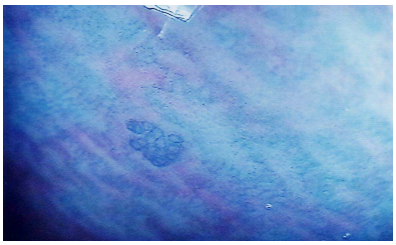
A study was carried out to select optimal values for the constant parameters of the model. The influence of each parameter in the segmentation is depicted in figure 4, and the used values are shown in table 1. The segmentation is improving from left to right (figures on the right side show less false positives). Nevertheless, it is visible a segmentation curve that delimits an open region, which is a false positive resulting from the two-phase nature of the model. We noticed that the number of iterations can be lower for small images and still achieve good results, but should be higher for larger images in order to obtain a superior segmentation quality. In the latter case, typically more common in real systems, it makes even more sense executing on the GPU.

For measuring the performance of the parallel algorithm on the GPU, execution times were obtained for different input data sizes and compared with MATLAB and sequential C implementations running on CPUs. Figure 5 shows a speedup of $43\times$ regarding to sequential C and of $161\times$ regarding to Matlab for a 1 Mega pixel image. These speedups increase to $45\times$ and $168\times$ respectively, for a 4 Mega pixel image.



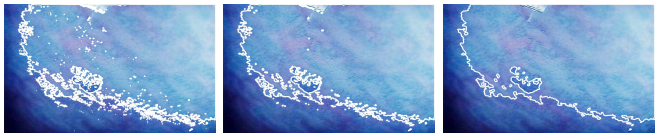
(a) (b) (c) (d)

Fig. 3. (a) and (c) depict input images; (b) and (d) show output segmented images (the parameters are indicated in table 1).



(a) Original image of a stained colon surface with an ACF

(b) Varying λ



(c) Varying θ

(d) Varying β

(e) Varying δt

(f) Varying $iterno$

(g) Varying $updtFT$

Fig. 4. Study of segmentation parameters, by changing one of its values and fixing the others, with D values from table 1, for an endoscopic image with ACF, and size 1182×714 pixels. (see L, C and R in table 1 for the corresponding values).

5. RELATED WORK

Although specific ACF segmentation on GPUs is, to the best of our knowledge, inexistent, there is active contour related

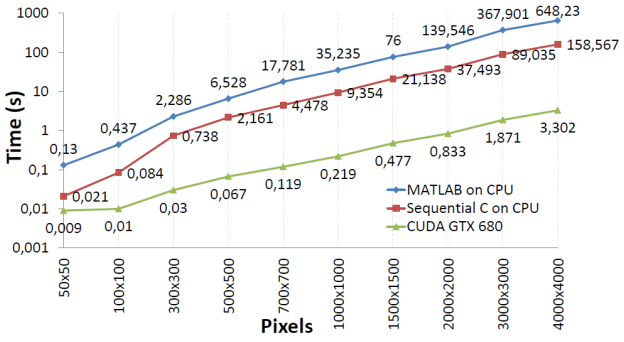


Fig. 5. Execution time of the algorithm for the various input data sizes and for the 3 different implementations, using $iterno = 500$ and $updtFT = 5$.

work done on GPUs. Zoltan *et al.* [9] implemented a color diffusion model as external energy for the active contour, which has the advantage of being able to process color images. Ali and Madabhushi [14] developed a multiple level set-based hybrid active contour framework which is capable of segmenting overlapped objects. While He and Kuester [13] used a gradient vector flow approach to compute the external force of the active contour, Li *et al.* [10] presented an optimal parametric active contour method based on Fourier descriptors, and Gilles *et al.* [11] developed a statistical region-based active contour model. In all these works the used algorithms have the initial contour problem. Pryor *et al.* [12] used a deterministic observer model based on nonparametric implicit curves that applies to dynamically evolving contours, which is not a requisite in ACF segmentation.

Our contribution to the ACF segmentation problem is a parallel approach based on a fast minimization of the active contour model [19], with the initialization conditions and local minima problems eliminated. Also, we propose a signal processing GPU approach that shows very fast execution times and suits the development of real-time systems. Clearly this can lead to superior/faster diagnosis accuracy and hopefully to the increase of human life expectancy.

6. CONCLUSION

We proposed a parallel algorithm for segmenting ACFs on GPUs that it is capable of processing four 1 Mega pixel frames per second. We analysed the influence of different parametrization levels using the GPU, which allowed a faster calibration of the algorithm. Consequently, we were able to propose optimal segmentation configurations. These results encourage the development of a real-time procedure to increase the success level of *in vivo* diagnosis.

Future work will incorporate a post-processing method to remove open regions from the segmentation result. Additionally, we are developing new algorithms capable of segmenting very small polyps that are equally CRC precursors.

7. REFERENCES

[1] American Cancer Society, "Colorectal cancer facts & figures 2011-2013," Tech. Rep., Atlanta: American Cancer Society, 2011.

[2] M. Zavoral, S. Suchanek, F. Zavada, L. Dusek, J. Muzik, B. Seifert, and P. Freic, "Colorectal cancer screening in Europe," *World J Gastroenterol*, vol. 15, no. 47, pp. 5907–5915, 2009.

[3] E. T. Hawk, A. Umar, and J. L. Viner, "Colorectal cancer chemoprevention — an overview of the science," *Gastroenterology*, vol. 126, no. 5, pp. 1423–1447, 2004.

[4] F. A. Orlando, D. Tan, J. D. Baltodano, T. Khoury, J. F. Gibbs, V. J. Hassid, B. H. Ahmed, and S. J. Alrawi, "Aberrant crypt foci as precursors in colorectal cancer progression," *Journal of Surgical Oncology*, vol. 98, no. 3, pp. 207–213, 2008.

[5] R. P. Bird, E. A. McLellan, and W. R. Bruce, "Aberrant crypts, putative precancerous lesions, in the study of the role of diet in the aetiology of colon cancer," *Cancer surveys*, vol. 8, no. 1, pp. 189–200, 1989.

[6] D. P. Hurlstone, S. S. Cross, A. J. Shorthouse, S. Brown, I. Adam, and A. J. Lobo, "Rectal aberrant crypt foci identified using high-magnification-chromoscopic colonoscopy: biomarkers for flat and depressed neoplasia," *The American Journal of Gastroenterology*, vol. 100, no. 6, pp. 1283–1289, 2005.

[7] I. N. Figueiredo, P. N. Figueiredo, G. Stadler, O. Ghattas, and A. Araújo, "Variational image segmentation for endoscopic human colonic aberrant crypt foci," *IEEE Transactions on Medical Imaging*, vol. 29, no. 4, pp. 998–1011, 2010.

[8] G. Falcao, V. Silva, L. Sousa, and J. Andrade, "Portable LDPC decoding on multicores using OpenCL," *IEEE Signal Processing Magazine*, vol. 29, no. 4, pp. 81–109, 2012.

[9] G. Zoltan, D. Stoica, and M. Ivanovici, "Colour diffusion model acceleration on GPUs," *International Conference on Optimization of Electrical and Electronic Equipment*, pp. 1429–1435, 2012.

[10] T. Li, A. Krupa, and C. Collewet, "A robust parametric active contour based on fourier descriptors," *IEEE International Conference on Image Processing*, pp. 1037–1040, 2011.

[11] P. Gilles, S. Domas, R. Couturier, and N. Bertaux, "GPU implementation of a region based algorithm for large images segmentation," *IEEE International Conference on Computer and Information Technology*, pp. 291–298, 2011.

[12] G. Pryor, T. ur Rehman, S. Lankton, P. A. Vela, and A. Tannenbaum, "Fast optimal mass transport for dynamic active contour tracking on the GPU," *IEEE Conference on Decision and Control*, pp. 2681–2688, 2007.

[13] Z. He and F. Kuester, "GPU-Based Active Contour Segmentation Using Gradient Vector Flow," in *Proceedings of the Second international conference on Advances in Visual Computing - Volume Part I*, Berlin, Heidelberg, 2006, ISVC'06, pp. 191–201, Springer-Verlag.

[14] S. Ali and A. Madabhushi, "Graphical processing unit implementation of an integrated shape-based active contour: Application to digital pathology," *Journal of Pathology Informatics*, vol. 2, no. 2, pp. 13, 2011.

[15] T. P. Chen and Y.-K. Chen, "Challenges and opportunities of obtaining performance from multi-core CPUs and many-core GPUs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 613–616.

[16] R. P. Bird, "Observation and quantification of aberrant crypts in the murine colon treated with a colon carcinogen: Preliminary findings," *Cancer Letters*, vol. 37, no. 2, pp. 147–151, 1987.

[17] S. J. Alrawi, M. Schiff, R. E. Carroll, M. Dayton, J. F. Gibbs, M. Kulavlat, D. Tan, K. Berman, D. L. Stoler, and G. R. Anderson, "Aberrant crypt foci," *Anticancer Research*, vol. 26, no. 1A, pp. 107–119, 2006.

[18] D. G. Adler, C. J. Gostout, D. Sorbi, L. J. Burgart., L. Wang, and W. S. Harmsen, "Endoscopic identification and quantification of aberrant crypt foci in the human colon," *Gastrointestinal Endoscopy*, vol. 56, no. 5, pp. 657–662, 2002.

[19] X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran, and S. Osher, "Fast global minimization of the active contour/snake model.," *Journal of Mathematical Imaging and Vision*, vol. 28, no. 2, pp. 151–167, 2007.

[20] T. Chan and L. Vese, "Active Contours Without Edges," *IEEE Transactions on image processing*, vol. 10, no. 2, pp. 266–277, 2001.

[21] "NVIDIA CUDA C programming guide," NVIDIA Developer Technology, 2012.