J Supercomput
DOI 10.1007/s11227-011-0597-1

A GPU-based implementation of the MRF algorithm in ITK package

Pedro Valero · José L. Sánchez · Diego Cazorla · Enrique Arias

Abstract The analysis of medical image, in particular Magnetic Resonance Imaging (MRI), is a very useful tool to help the neurologists on the diagnosis. One of the stages on the analysis of MRI is given by a classification based on the Markov Random Fields (MRF) method. It is possible to find in the literature several packages to carry out this analysis, and of course, the classification tasks. One of them is the Insight Segmentation and Registration Toolkit (ITK). The analysis of MRI is an expensive computational task. In order to reduce the execution time spent on the analysis of MRI, parallelism techniques can be used. Currently, Graphics Processing Units (GPUs) are becoming a good choice to reduce the execution time of several applications at a low cost. In this paper, the authors present a GPU-based classification using MRF from the sequential implementation that appears in the ITK package. The experimental results show a spectacular execution time reduction being the GPU-based implementation up to 118 times faster than the sequential implementation included in the ITK package. Moreover, this result is also observed by reducing the total power consumption in a significant amount.

P. Valero (✉)
Albacete Research Institute of Informatics, Avda. España s/n, 02071 Albacete, Spain
e-mail: pedro.valerolara@uclm.es

J.L. Sánchez · D. Cazorla · E. Arias
Computing Systems Dept., University of Castilla-La Mancha, Avda. España s/n, 02071 Albacete, Spain

J.L. Sánchez
e-mail: jose.sgarcia@uclm.es

D. Cazorla
e-mail: diego.cazorla@uclm.es

E. Arias
e-mail: enrique.arias@uclm.es

Published online: 30 March 2011

Keywords Magnetic resonance imaging · Markov random fields · Insight toolkit · Graphics processing units

1 Introduction

The main goal of the Spanish project *Alztools* (TSI-020110-2009-362) is to develop a High Performance Computing tool for the analysis of Magnetic Resonance Imaging (MRI) from the brain in order to prevent Alzheimer diseases.

One of the most useful algorithms used on medical image analysis is the so-called Markov Random Fields (MRF). In the literature, the MRF algorithm has been applied in different areas as, for instance, speech recognition [1], analysis of satellite images, etc. Focusing on the analysis of MRI according to the objectives of *Alztools* project, there is a set of widely used tools by the neurologists: SPM, Freesurfer, FSL, 3DSlicer, and ITK, among others. In the *Alztools* project, the ITK [2] library has been chosen due to the fact that is an opensource under the GNU license, it is maintained and updated with more efficient algorithms, and it can be integrated with other libraries, as for example, 3DSlicer, etc. Moreover, this library is totally integrated and is widely used in the medical community. Even this community participates in the maintenance and development of ITK, under the direction of the National Library of Medicine and National Institutes of Health. Evidently, the MRF classification algorithm, the subject of this paper, is present in ITK.

Current GPUs are an appropriate parallel platform in order to accelerate this type of applications, due to their characteristics, emphasizing that these devices have a low cost/performance ratio. In fact, current GPUs are beginning to be used in HPC environments. In order to obtain HPC tools, it is very important to reduce their response time. Therefore, optimizing and/or parallelizing the most time consuming operations is becoming a priority in the medical image processing. The main result of this work is the significant reduction of the execution time of ITK MRF algorithm.

The paper is structured as follows: Sect. 2 briefly introduces the classification process in the image analysis and describes the sequential implementation of the classification using Markov Random Fields included in the ITK package. Section 3 shows the great computational capacity of the current GPUs. Using the sequential code as the starting point, in Sect. 4, the GPU-based implementation of the MRF-based classification algorithm is introduced. Section 5 shows the experimental results and the analysis of performance. Finally, Sect. 6 outlines the conclusions and future work.

2 MRF-based classification in the ITK package

Given an image, multivariate data (feature vectors) are observed at respective pixels. The main objective of the feature vector $(v_1, v_2, \ldots, v_n)$ is to describe the image in terms of several, n, attributes. Image classification is a problem of classifying pixels into several homogeneous regions by learning the feature vectors and the adjacency relationships of the pixels in the image, representing an important and fundamental problem in image pattern analysis [3].

Image classification analyzes the properties of various image features and organizes data into categories [4]. In image classification, normal distributions are frequently used for analyzing multivariate data in a feature space, and Markov Random Fields (MRFs) are used for modeling the distribution of categories in the image [5]. It is usually assumed that the category labels follow the MRF. The estimation of parameters specifying the MRF is not an easy task because the probability distribution cannot be expressed in a closed form. Hence, the pseudo-likelihood is frequently used for this purpose. The key issue is the estimation of pixel labels of test data. Computer-intensive methods [6] can be used for the estimation, but the implementation is often difficult because of computational complexity.

MRFs are probabilistic models that use the correlation between pixels in a neighborhood to decide the object region or category. In order to consider the neighborhood information, a simple approach consists in estimating the joint posterior probability for the whole image's labeling configuration. The size of the images can become a problem even not considering high resolution. For example, for a 620×540 image, there will be 334,800 pixels needed to be considered. After determining features, each pixel will have a corresponding feature, which generally will be larger than 1. The huge amount of information to manage will make the joint of the whole image hard to compute.

Different alternatives have been used to model MRF. We focus on that included in the ITK package, contributing at image classification process.

MRF is implemented in ITK at *itk::Statistics::MRFImageFilter* method which uses the maximum a posteriori (MAP) [8] estimates for modeling the MRF. The object traverses the data set and uses the model generated by the Mahalanobis distance classifier [9] to get the distance between each pixel in the data set to a set of known classes, updates the distances by evaluating the influence of its neighboring pixels (based on a MRF model), and finally, classifies each pixel to the class which has the minimum distance to that pixel (taking into account the neighborhood influence). The energy function minimization is done using the iterated conditional modes (ICM) algorithm [6].

The main use of the *itk::Statistics::MRFImageFilter* method is for refining an initial classification by introducing the spatial coherence of the labels. The user should provide two images as input. The first image is the one to be classified while the second image is an image of labels representing an initial classification. Algorithm 1 sums up the sequential implementation of the CMRF algorithm implemented in ITK.

3 Graphics Processing Units

Current Graphics Processing Units (GPUs) consist of a high number (e.g., 128–240) of fragment processors with high memory bandwidth. They can offer $10\times$ higher main memory bandwidth and use data parallelism to achieve up to $10\times$ more floating point throughput than the CPUs [7].

GPUs are traditionally used for interactive applications, and are designed to achieve a high rasterization performance. However, their characteristics have led to the opportunity to other more general applications to be accelerated in GPU-based

Algorithm 1 Pseudocode of CMRF implemented in ITK

Function *I_Class* = `MRFImageFilter`(*I_MRI*, *I_Label*, *Classes*, *ConVar*, *Window*)
Inputs:
I_MRI : *MRI image*
I_Label : *Labeled image*
Classes : *Classes considered to classify*
ConVar : *Total pixels changed at the previous iteration*
Window : *Size of the window*
Output:
I_Class : *Classified image*

```
1: while ( do(iter_actual < MAX_ITER) and (Error > MAX_ERROR))
2:     while ( doNo_Final_Image)
3:         Compute influence of each Class on central pixel of the Window over I_MRI.
4:         Compute the influence of each Class of all the pixels of the Window around the
           central pixel of the I_Label.
5:         Compute the Mahalanobis Distance of the central pixel of the Window
6:         Compute the new label of the central pixel
7:     end while
8:     Compute Error
9: end while
```

platforms. This trend is called General Purpose Computing on GPU (GPGPU). These general applications must have parallel characteristics and an intense computational load to obtain a good performance.

To assist in the programming tasks of these devices, the GPU manufacturers, like NVIDIA or ATI, have proposed new languages or even extensions for the most common used high level programming languages. As an example, NVIDIA proposes CUDA, which is a software platform for massively parallel high-performance computing on the company powerful GPUs.

In CUDA, the calculations are distributed in a mesh or grid of thread blocks; all thread blocks are the same size (number of threads). These threads run the GPU code, known as kernel.

As mentioned in Sect. 2, image processing based on MRF has high computational complexity. Therefore, it is reasonable that GPUs are used for running these kinds of applications.

4 A GPU-based implementation of the classification using MRF method in the ITK package

In this paper, a coarse-grain implementation of the sequential CMRF algorithm is considered. Thus, the parallel implementation using a GPU consists in carrying out all the operations over the image at the same time.

According to step 5 of the sequential implementation, several operations are carried out in order to compute the Mahalanobis distance, and then to decide if the value of the central pixel has to change or not. These operations are considered for any window and they have to be done in a determined order.

Algorithm 2 Pseudocode of GPU-CMRF algorithm

Function *I_Class* = `MRFImageFilter`(*I_MRI*, *I_Label*, *Classes*, *ConVar*, *Window*)
Inputs:
I_MRI : *MRI image*
I_Label : *Labeled image*
Classes : *Classes considered to classify*
ConVar : *Number of pixels changed at the previous iteration*
Window : *Size of the window*
Output:
I_Class : *Classified image*

1: Compute influence of each *Class* on central pixel of the *Window* over *I_MRI*.
2: **while** (**do**(*iter_actual* < *MAX_ITER*) and (*Error* > *MAX_ERROR*))
3: Compute the influence of each *Class* of all the pixels of the *Window* around the central pixel of the *I_Label* and Compute the Mahalanobis Distance of the central pixel of the *Window*.
4: Compute the new label of the central pixel
5: Compute Error
6: **end while**

The sequential version imposes an order in the operations. Thus, the parallel implementation needs to launch different kernels, steps 1, 3, and 4 on Algorithm 2, launching as many threads as the number of windows in the image. The number of threads is given by the image size and the window size.

Algorithm 2 is similar to the one presented in Algorithm 1, but without the internal *While* loop.

5 Performance evaluation

In this section, a performance analysis will be carried out considering the sequential implementation of the CMRF algorithm included in ITK package and the GPU-based CMRF implementation presented in this work.

We show results of time and power consumption. As shown below, an important execution time reduction is obtained through the use of GPU-based computing platform. However, current GPUs are composed of a large number of transistors, and so they suffer from higher power consumption requirements. Therefore, it is necessary to develop energy-efficient GPU codes and as a consequence power consumption becomes an essential metric in this kind of studies.

In this work, power measurements have been performed using a system composed of two computers. The first one (host) is the computer in which the MRF algorithms are running and which energy we want to measure. The second one is used in order to receive and process the energy data.

We use a sensor (Allegro Microsystems Inc. A1301) capable of translating magnetic changes into a proportional voltage level to work in a comfortable way. The sensor output is tied to the analog port in a microcontroller (Microchip PIC12F683). This microcontroller is the responsible of sampling the voltage data and sending it to the user. The data transfer rate of our device is 115200 bauds using RS232 protocol.

For the serial communication, Future Technology Devices International Ltd. FT232 chip has been used. It allows us to change the RS232 environment to the USB environment. The data is received through a virtual COM port which is created when the installation of the sensor node in the host computer is completed. The software used to collect the data is the Eltima software RS232 datalogger.

Finally, once the data for the intensity and voltage are known, we can obtain the power consumption applying the equation $P = V \times I$ ($V = 230$ V).

The algorithms have been run in the host computer with the following features:

- CPU: Intel Core 2 Quad at 2.66 GHz and 4 GB of main memory.
- GPU: GTX 285 with 240 cores and a main memory of 1 GB.

Different case studies have been considered according to the following parameters:

- Resolution of MRI: 0.488 mm, 0.5 mm, 1 mm, and 2 mm.
- Image size: 628×544 pixels if the resolution is 0.488 mm, 364×436 pixels if the resolution is 0.5 mm, 182×218 pixels if the resolution is 1 mm, and 91×109 pixels if the resolution is 2 mm.
- Window size: 3×3, 5×5, and 7×7.
- Number of classes: 3 and 4.

Table 1 sums up the results in terms of execution time, including the communication time between the CPU and GPU memory spaces, by considering the different resolutions (0.488 mm first row, 0.5 mm second row, 1 mm third row and 2 mm fourth row) of MRI.

Figure 1 shows the speedup considering the different resolutions of MRI, for 3 and 4 classes. According to the experimental results, the execution time has been dramatically reduced by using the GPUs.

These results have been obtained after several optimizations: usage of shared memory, loop unrolling, and reduction of the number of records needed for each kernel.

From the previous general conclusion, the following conclusions can be outlined:

- As the resolution of the MRI increases, that is, the size of the image increases in terms of number of pixels, the execution time evidently also increases. Thus, the use of GPUs provides better performance in terms of speedup considering a high

Table 1 Execution time of the sequential and parallel algorithm on a GPU GTX 285

Time (sec.) for sequential algorithm						Time (sec.) for parallel algorithm					
3 Classes			4 Classes			3 Classes			4 Classes		
3×3	5×5	7×7	3×3	5×5	7×7	3×3	5×5	7×7	3×3	5×5	7×7
3.61	12.84	41.60	3.90	16.27	26.78	0.16	0.25	0.35	0.18	0.31	0.41
0.90	2.90	19.00	7.24	12.51	17.09	0.09	0.15	0.19	0.11	0.17	0.22
0.24	0.6	1.31	0.27	1.71	1.46	0.05	0.06	0.08	0.05	0.08	0.09
0.08	0.28	0.40	0.11	0.16	0.27	0.04	0.06	0.06	0.04	0.05	0.06

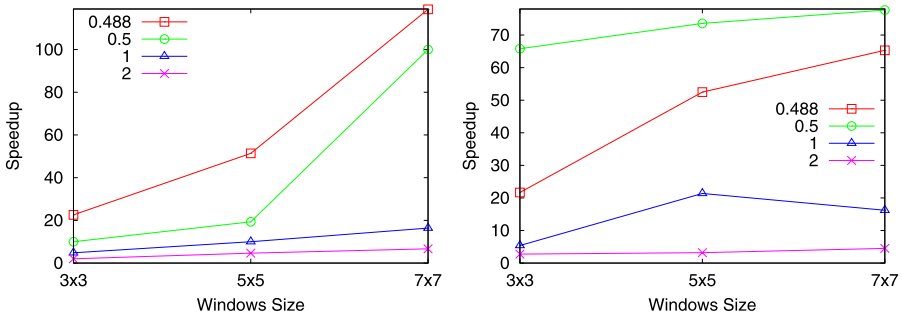

Fig. 1 Speedup of the GPU parallel implementation on a GPU GTX285 considering 3 classes (*left*) and considering 4 classes (*right*)

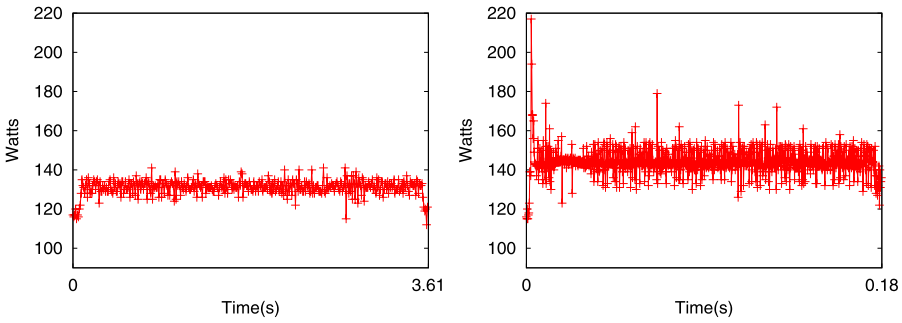

Fig. 2 Energy consumption. Sequential (*left*) and parallel (*right*) implementation

resolution MRI. From the medical point of view, a high resolution MRI is more interesting due to the fact that it provides better accuracy.

– On the other hand, as the size of window considered on CMRF method increases, the execution time also increases, but again the accuracy increases. Newly, the use of the GPUs is benefited with respect to the sequential implementation as the problem size increases.

To sum up, from the medical point of view it is interesting to deal with high resolution MRI and a high size of window due to the fact that more accuracy is obtained. Under these conditions, but no constraints, the GPU-based implementation is up to 118 times faster than the sequential implementation.

Figures 2 and 3 show power consumption results. These results correspond to the following scenario (first case in Table 1): image of 628×544 pixels, 3 classes, and window 3×3. In other cases, the results follow the same tendency. When GPU is used, there is a slight increase in the power consumption. However, the parallel implementation consumes less energy than sequential version due to the significant execution time reduction. This can be observed much better in Fig. 3, where we show the average power consumption and execution time. Through this data, in an approximate way, the difference in terms of power consumption between both versions is 94.53%.

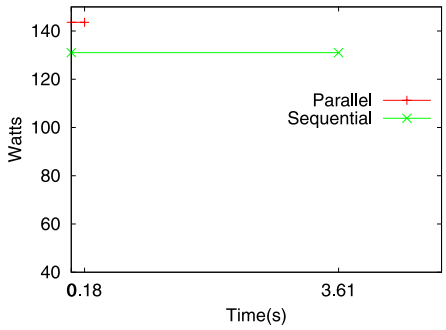

Fig. 3 Power consumption and execution time for the sequential and parallel implementations

6 Conclusions and future work

Image analysis is becoming an important discipline applied to different areas in science and engineering. Our research is focused on the analysis of Magnetic Resonance Imaging (MRI). In this context, several MRI analysis packages have appeared. Among them, one of the most widely used is the Insight Segmentation and Registration Toolkit (ITK).

An important task on image analysis is the classification of images. In ITK, the Classification using Markov Random Fields (CMRF) is used. In this work, a GPU-based implementation of CMRF algorithm in ITK has been developed.

According to the experimental results, it is remarkable the spectacular execution time reduction by the use of a GPU-based platform, reaching a speedup close to 118, and consuming less energy for its execution.

At the moment, we have only used 2D images. Even so, the performance obtained encourage to continue not only considering 3D images, but also considering new algorithms useful by the neurologist as Hiden Markov Model, Nonlinear Registration, etc.

References

1. Gravier G, Sigelle M, Chollet G (2000) A Markov random field model for automatic speech recognition. In: Proceedings of the international conference on pattern recognition (ICPR'00), vol 3. IEEE Computer Society, Los Alamitos, p 3258. ISSN:1051-4651
2. ITK - Segmentation & Registration Toolkit. http://www.itk.org/
3. Mardia KV (1988) Multi-dimensional multivariate Gaussian Markov random fields with application to image processing. J Multivar Anal 24:265–284
4. Duda RO, Hart PE, Stork DG (2000) Pattern classification, 2nd edn. Wiley-Interscience, New York
5. Li SZ (2000) Modeling image analysis problems using Markov random fields. In: Handbook of statistics, vol 20. Wiley, New York, pp 1–43
6. Besag J (1986) On the statistical analysis of dirty pictures. J R Stat Soc B 48:259–302
7. Feng W-c, Manocha D (2007) High-performance computing using accelerators. Parallel Comput 33:645–647
8. DeGroot M (1970) Optimal statistical decisions. McGraw-Hill, New York
9. McLachlan GJ (1992) Discriminant analysis and statistical pattern recognition. Wiley-Interscience, New York