MATLAB MEDICAL IMAGES CLASSIFICATION ON GRAPHICS PROCESSORS

Ramona DIN

Due to their massively parallel hardware design, graphic processors can easily beat ordinary CPUs in applications which involve large amount of data. Considering their great potential, the objective of this paper is to continue previous work and optimize the speed and efficiency of texture and fractal analysis, as used for medical images classification processes for early skin cancer detection. The images are classified based upon Haralick features and fractal geometry, which both proved to be traditionally computationally expensive.

Keywords: medical image processing, parallel computing, content based image retrieval, texture analysis, fractal analysis, graphics processing unit

1. Introduction

We can nowadays easily identify information retrieval principles at the core of almost all digital information related processes, as it forms the foundation of any search engine. With a great impact on present day large scale systems and services, information retrieval (IR) deals with searching and managing huge volumes of data, mainly large collections of text, images, music and other human-language data representations. Millions of people are relying on information retrieval systems to accomplish their professional, academic or personal targets: enterprise platforms, social, mobile, or web search engines, digital libraries, medical simulations, computer aided diagnosis, image, news, video, or music search – just to name few areas for which retrieval techniques are vital. As a common basis, all engage at least one form of modern information retrieval, overtaking traditional database searching style. The most sophisticated information retrieval systems easily deal with unstructured data, i.e. data which does not have semantically clear, easy for a computer to interpret, structure. As a challenge for many years now, but emulating the latent linguistic structure of human language, this represents exactly the opposite of structured data, the canonical example of which is a classic relational database. Although its origins are to be found in text processing algorithms, one of the most challenging and pressing areas of IR still remains image retrieval. Over the last decade, image retrieval systems were designed to deal with image collections as personal pictures, domain-specific collections (e.g. medical databases, satellite images, mathematical, physical or biological simulations), enterprises' image collections,

archives, and last, but definitely not least, with the world wide web. All of them seek for a high degree of user satisfaction and thus, each focuses on different features such as personalization, data display, browsing, processing speed, or accuracy.

One considerable application area for image retrieval techniques is for medical databases. Medical imaging produces great amounts of data which, once investigated and classified, might be extremely valuable for future diagnoses. The approach that most of existing medical information retrieval systems considered so far was to analyze images' content, extract relevant data, and exploit it in such a way that it allows a meaningful classification. Most of the systems were designed for computer aided diagnosis, but currently their features extends also to assist physicians in treatment of diseases, in choosing or defining an adequate therapy, or for surveying patient's evolution during the treatment [1], [2]. Depending upon particular tasks, the classification process might use different image features with various properties. Among the most common are the textural properties, color, shape, or fractal properties. However, unstructured data, heterogeneous tissues, undefined shapes, and the noise presence negatively influence automated analysis. In order to address these challenges, one can either simplify the scope of medical analysis or pre-process the image so that he exploits some kind of a priori information about the imaged structured. This can refer to anatomical knowledge (shape, position, gray levels) or to its statistical properties [3]. Nevertheless, the medical image processing is still a heavy task as the computing time increases with the image size and resolution. Fortunately, from a technically perspective, the computation can be optimized and partitioned into parallel units of work, and then each executed separately. In the beginning of parallelization era, the parallelization was achieved at the virtual levels, leveraging and synchronizing bunches of parallel threads. Once the multi-core processors became available on the market, a great number of parallel algorithms were readjusted to exploit CPU's multicores. The application of parallelism and distributed techniques introduced improvements for a large number of algorithms, some of them applied for search systems as well. Although parallel computing algorithms are, most of the time, more complex than their sequential equivalents, they have always had the same major objective: achieving a performance which is superior to the sequential processing one, whether this means identifying a solution in less time, simply identifying a better solution, process a larger set of data or even solving high complexity problems.

The past few years, microprocessor design has followed two separate paths: on the one hand, there were the *multi-core* multiprocessors (e.g. IntelCore i7 is a microprocessor with four processing cores, each of them implementing a complete x86 instructions set); these were offering support for multiple hardware threads and were mainly used for improving the execution of sequential

algorithms. A performance improvement of 100% could be achieved only by doubling the number of processing cores within a microprocessor; therefore the improvements were highly limited by hardware. On the other hand, the *many-core* microprocessors (e.g. graphics processors) developed mainly by the video-card manufacturers using a different design strategies, were trying to achieve a high level of hardware parallelization. The technology used allowed the manufacturers to build these microprocessors with a considerable number of cores (e.g. NVIDIA's GTX470 has 448 cores), each of them being highly parallelized at the hardware level, massively reducing hardware constraints as opposed to the ones above described. Since 2003, microprocessors having multiple execution cores, especially graphics microprocessors, have managed to achieve performances which are far better than those of multi-core microprocessors and this has been well documented in literature [4]. Moreover, the performance improvement factor can easily exceed the level of 100% - maximum allowed by CPU's hardware design.

There are fundamental design differences between the two presented microprocessors types, differences which lead to performances of up to 20 times better in the case of not optimized algorithms [5] and up to 400 times better in the case of specialized algorithms [6]. More than that, a hardware structure which is specialized in the rapid execution of sequential implementations, allows central processing units (CPU) to use sophisticated logical mechanisms in order to be able to execute a single thread's instructions in parallel. In order to reduce the execution time, these require a very large cache memory and high bandwidth, both of which are currently hardware constrained. Many-core processors, especially graphics processing units (GPUs), have led the race of performance since 2003 [7]. While the performance improvement of CPUs has significantly slowed, the GPUs conquer the market, improving relentlessly. As of 2009, the ration between many-core GPUs and multi-core CPUs for peak floating-point calculation throughput is about 10 to 1. Even more, graphic chips have been operating at almost 10 times the bandwidth of contemporary CPU chips. G80 processors and their successors allow CUDA programs to run without going through the graphic interface at all [6]. Instead, a brand new general-purpose parallel programming interface on the chip serves all requests of CUDA enabled programs. Nowadays, the graphic processor is already a market old-timer. Associating its great potential with the reality that parallel algorithms as a concern of programming paradigm, have as long of a tradition as the one of sequential algorithms [8], this paper aims to introduce a novel approach for content-based medical image retrieval based upon texture and fractal features computed on graphical platforms. In [3] the authors proved that both texture and fractal analysis may be successfully applied for medical image classification for early detection of skin cancer. This paper aims to continue the previous work and optimize the

processing speed and efficiency by implementing all the algorithms and procedures on a parallel processing platform. The results will be compared and presented against already published CPU results [3].

2. Image retrieval systems - state of the art

The early image retrieval systems were based upon metadata or so-called annotations. Over the last decade, researchers have built numerous image annotation-based and automatic annotation features for various image retrieval systems. J. Li et al proposed automated annotation of pictures with a few hundreds of words and using Markov models in [9]: their classification process chooses a set of classes an image might belong to based on a set of annotations defined using statistically salient words for given images. In some other researches, annotations are considered to be translations in between a predefined set of words and images [10]. Hierarchical statistical techniques which associate images to words (i.e. automatically annotates images) has been also proposed in [13], [12], and [11]. Generative language models have been used for the task of image annotation in [14], [16]. Closely related, another approach was to involve coherent language models, exploiting word-to-word correlations to strengthen annotation decisions [15]. The proposed system automatically computes annotation length for a given image, hence using annotations variable in size. The authors attempted, this way, to provide a more accurate description, eliminating irrelevant words. They have proved that automatically determined annotation length might improve the accuracy of the image retrieval system, but only for a relatively small range of text queries. In order to improve system's performance, they have introduced an active learning module, reducing the number of annotated images. Their empirical studies have shown that the results were substantially more effective than using a simple random sampling approach.

All the annotation strategies discussed so far model visual and textual features separately prior to association and none discusses or addresses the problem of semantic gap. In [17], the authors tried to use latent semantic analysis (LSA), direct match, and annotation by inference (PLSA) for both annotations and visual features. The LSA model has been previously used to identify semantically meaningful subspaces in the visual-textual feature space, but automated image annotation still remained a difficult question. Humans learned to associate multiple viewpoints to static images. The association of words and blobs become truly meaningful only when blobs isolate objects well. Moreover, how exactly our brain performs this association is still unclear. While Biology tries to answer this fundamental question, researchers in information retrieval tend to take a pragmatic stand in that they aim to build retrieval and annotation systems that have practical significance [18].

Text-based image retrieval systems were a great step forward in digital information processing, but they had few significant drawbacks. Firstly, they involve significant amounts of manual work to be done: each image needs to be manually annotated with relevant data by a human being. Beside the fact that this requires effort, it is time consuming and it cannot be automated according to the current approach. On the same note, different people may describe the content of an image in different ways, which limits the search process to metadata used in description. Moreover, most of the times, textual description is not capable of comprising visual contents and their meanings; it may be the case that for some images there is something that can not be conveyed by words at all. Even further, the most intriguing challenge about multimedia retrieval is when the query itself presents as a multimedia excerpt. For instance, doctors may use medical images for searching similar previous cases before deciding on a diagnose, tourists may use pictures trying to identify places they visit, meteorologists may have satellite images as inputs for their weather prediction systems. Modern image retrieval systems aim to accommodate all these and, the most successful approach so far is based upon image features extraction. Weather the retrieval is good or not is not a matter of subjectivity anymore; it depends on the selected features and their extraction accuracy. In [19] image features have been classified by semantic hierarchy into middle level features and low level features. The last class includes color, texture, and inflection, whereas the former refer to shape and objects' features. In medicine, computing textural properties and applying them for image retrieval is not a novelty: Sutton and Hall proposed pulmonary diseases' classifications based on them, Harms et al have also used a combination between texture and color features to identify malignancy in blood cells, Insana et al applied textural features to estimate tissue scattering parameters in ultrasound images [3]. Recently, fractal geometry has been also exploited to investigate epithelial complexity and malignancy of various tumors; this has been considered as a promising approach for early diagnosis in gastric tumors [20], breast, and pulmonary cancer [21], [22].

Several methods have been described in literature for texture feature extraction. One might use statistical, structural, model-based and transformation information, wherein a common technique bases on Gray Level Co-occurrence Matrix (GLCM), proposed by Haralick in 1973 [23]. After computing the co-occurrence matrix, Haralick suggested 14 most useful statistical features: angular second moment, contrast, energy, correlation, variance, inverse difference moment, sum average, sum entropy, entropy, difference variance, difference entropy, information measure of correlation 1, information measure of correlation 2, and maximum correlation coefficient. GLCM is a useful tool in image processing area and texture analysis; however it is a heavy and computational intensive method, almost always being time-consuming. In [24], the authors

presented an example of GLCM calculation for medical applications. The computation tine for an image of 5000x5000 pixels was about 350 seconds using one CPU running at 2400MHz. Inspecting where the bottleneck lies, they found out that 75% of total time was spent calculating the GLCM matrix, 19% extracting features and the rest for the classification. Previously, there were many researches focused on accelerating the computation of GLCM and more then a few methods have been analyzed and tested by researchers. Using a linked list (Gray Level Co-occurrence Linked List – GLCLL) and storing just non-zero values [25], or Gray Level Co-occurrence Hybrid Structure – GLCHS [26], representing the image on 4 bits instead of 8, all were improvement suggestions already investigated, but, to the best of our knowledge, none didn't manage to achieve outstanding results.

Due to their massively parallel hardware design, GPUs for general-purpose can easily beat ordinary CPUs in applications which involve large amount of data. Historically, GPUs were born for being used in advanced graphics and videogames [6]; still, more recently interfaces have been built to interact with codes not related to graphical purposes. General-purpose computing on graphical processing units (GPGPU) is an emerging method for achieving high performance gains on various computing problems and scientific researches such as neural networks, medical research, database operations, information retrieval, and physics-based simulations. The higher number of cores and the more efficient processing of complex mathematical calculations on GPUs in comparison with CPUs, makes general purpose computation on graphics hardware (GPGPU) a fascinating new method for deploying algorithms [27]. Whereas lots of studies analyzed and proposed optimizations for CPU implementation, few tried to exploit GPU for performing such a task as medical image classification. Given their extremely high computing demands, image retrieval systems represent a very interesting challenge for GPUs. The next section presents some already studied techniques for content-based image retrieval, altogether with the architecture and integration of a suite of Matlab routines specially designed for GPU's platform. The fourth section of the paper presents our experimental results, followed by conclusions.

3. GPU Matlab implementations for image textural features extraction

3.1. Statistical methods to texture analysis

The most powerful statistical method for textured image analysis is based on features extracted from the GLCM, proposed by Haralick. The co-occurrence matrix considers the relationship between two neighboring pixels (one being the reference, the other the neighbor pixel). The image used to determine the co-

occurrence matrix is an image with a predefined number of gray levels. Let's assume G the number of gray levels for an image $I(x,y)$ with: $\begin{cases} 0 \le x \le N, \\ 0 \le y \le N \end{cases}$

The GLCM P_d^θ is defined for a displacement vector $d = (\Delta x, \Delta y)$ and direction θ as follows:

$$P_d^\theta(i,j) = Card\{((x,y),(t,v)) / I(x,y) = i, I(t,v) = j\}, \begin{cases} (x,y),(t,v) \in N\ xN \\ (t,v) = (x + \Delta x, y + \Delta y) \end{cases} \quad (1)$$

For each pixel, it has been considered a corresponding surrounding area, considering 8 main directions for each neighborhood when determining the co-occurrence matrix [28].

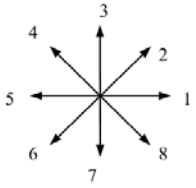

Fig. 3.1. Directions used in co-occurrence matrix calculus

Haralick texture features

From GLCM one can easily compute a number of texture features as defined by Haralick. All these features have a good discriminating power, being extremely useful for content-base image retrieval. For the current application in this paper, we have determined the entropy Ent, the contrast Con, the energy Eng, and the correlation factor Cor for a NxN co-occurrence matrix [28].

Entropy

The term originates from thermodynamics and it refers to the quantity of energy which is permanently lost in order to heat a reaction or other types of physical transformation. Within the current context, it denotes the degree of disorder, as opposed to homogeneity. Entropy's large values indicate uniform GLCM. It can be defined as:

$$Ent = -\sum_{i=1}^{N}\sum_{j=1}^{N} P_d(i,j)\log(P_d(i,j)) \quad (2)$$

Contrast

It measures the gray-level variations between the reference pixel and its neighbors; large values of contrast indicate large local variation of gray levels. In case of performing the computation on color-based images, the contrast is

determined by the difference in the color and brightness of one object and other object within the same area.

$$Con = \sum_{i=1}^{N}\sum_{j=1}^{N}(i-j)^2 P_d(i,j) \tag{3}$$

In case of $i = j$, the cell is on the diagonal and equality means the pixels are entirely similar to their neighbors, considering thus their weight null. The weights increase exponentially as $(i - j)$ increases.

Energy

It measures the degree distribution of gray levels; it has the highest value when this is either constant or periodic. The energy is a measure of the textural uniformity of the image and it can be defined as follows:

$$Eng = \sum_{i=1}^{N}\sum_{j=1}^{N}P_d(i,j)^2 \tag{4}$$

Correlation

It refers to the linear dependency of gray level values in the co-occurrence matrix, or, simply put, to the pixel's neighborhood influence over the entire image. A reference pixel may be uncorrelated to its neighbor (having correlation factor null), perfectly correlated (with a correlation factor of 1) or anywhere in between (with a correlation factor in between 0 and 1).

$$Cor = \sum_{i=1}^{N}\sum_{j=1}^{N}P_d(i,j)\frac{(i-\mu_i)(j-\mu_j)}{\sigma_i\sigma_j} \tag{5}$$

where $\mu_i, \mu_j, \sigma_i, \sigma_j$ are the means (6), respectively standard deviations (7) computed from the histogram, h_d, of the difference image [28]. In case of symmetrical GLCM, $\mu_i = \mu_j$ and $\sigma_i = \sigma_j$.

$$\mu_d = \frac{1}{N}\sum_{i=1}^{G}x_i h_d(x_i) \tag{6}$$

$$\sigma_d^2 = \frac{1}{N}\sum_{i=1}^{G}(x_i-\mu_d)^2 h_d(x_i) \tag{7}$$

3.2. Fractal dimension used in texture analysis

Fractal geometry has been previously applied to medical algorithms and became popular in modeling biological properties in image processing especially due to its capabilities of modeling complex shapes. By identifying contour

irregularities and computing fractal dimensions, one can discriminate between the shapes of malign over benign tumors. Experimentally, it was established that most of the information about the malignity of a tumor is contained in the contour of the tumor shape. Doubtful tumors are characterized by blurred contours which are changing by different threshold used to separate the tumor from background (image segmentation). The outline of each image was analyzed by estimating the global fractal dimension, the local fractal dimension and local connected fractal dimension. One of the most relevant concepts in fractal geometry is self-similarity: considering a bounded set A in a Euclidian n-space, one can state A is self-similar when it represents the union of N distinct and scaled copies of itself. The fractal dimension FD is defined as a function of N and the ratio between the original image and the scaled down copies:

$$FD = \frac{\log_2 N(r)}{\log_2 (1/r)} \tag{8}$$

The algorithm calculates a mean fractal dimension MFD (9) from individual values FD_i of different image contours detected exploiting the initial gray-level image:

$$MFD = \frac{1}{k} \sum_{i=1}^{k} FD_i \tag{9}$$

In (9), k represents a predefined value which specifies the number of iterations for contour extraction using a fixed neighborhood. The algorithm used to compute the fractal dimension was a version of box-counting, its steps been described in a previous paper [3]. Further, the current paper presents the implementation and integration of a new set of Matlab routines dedicated to graphic platforms.

3.3. System architecture

Fig. 3.2 presents the system architecture, all modules being implemented in Matlab and all, except from the one reading the database and saving back the results, are running on graphic processors.

In order to parallelize the code and execute it in on the GPU, we have rewritten previously used Matlab GPU routines using Jacket, a runtime platform that helps to connect the M language to the GPU. It offers support for specific data types – counterparts to CPU Matlab data types, and a set of GPU functions ranging from basic implementations to complex arithmetic or signal processing solving methods.

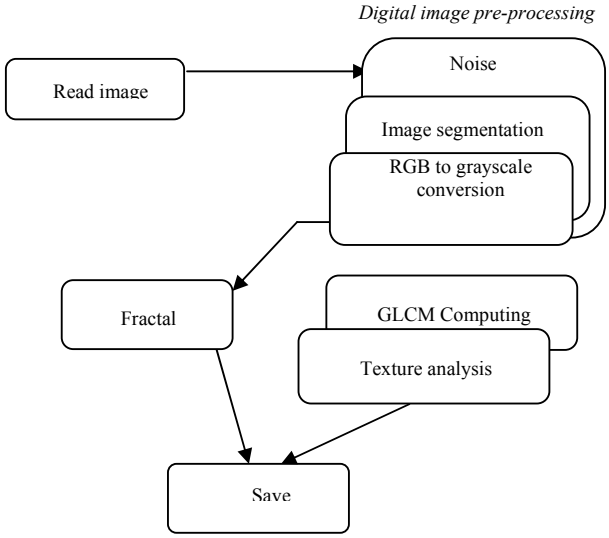

Fig. 3.2. System architecture

It provides automated memory management and compilation-on-the-fly, being in the same time transparent for programmer. Moreover, its capabilities extends to integrating any custom CUDA-C (and other programming languages) function into a suite of Matlab routines. The Fig. 3.3 presents the most relevant Matlab functions, based upon the architecture presented in Fig. 3.1. The entry point is represented by medicalImageAnalysis.m which transforms every image into its grayscale correspondent, computes the co-occurrence matrix, extracts four textural features (the entropy, the contrast, the energy, and the correlation), and computes the fractal dimensions. In order to create the gray-level co-occurrence matrix, pairs of horizontally adjacent pixels are evaluated within an image scaled to 8 levels. Therefore, there are 8 x 8 possible ordered combinations of values for each pixel pair and the result gets computed by adding the total occurrences of each combination. The user can choose any custom value for offsets or a default of *[0, 1]* is used when none was provided. If the corresponding neighbors defined by the offset fall outside the image boundaries, the implementation ignores border pixels. Also, pixels pairs are ignored if either of their values is NaN. The elemSum.m uses as input an intermediate matrix computed from the grayscale image, 0 and 1 – as the elements to be summed, and the output vector size, returning an array with values determined by accumulating 1 according to the subscripts from the first parameter. In order to compute the fractal properties, a box-counting algorithm has been implemented to run on the GPU platform. The implementation followed the example of a Matlab built-in function, using Jacket GPU-dedicated data types.

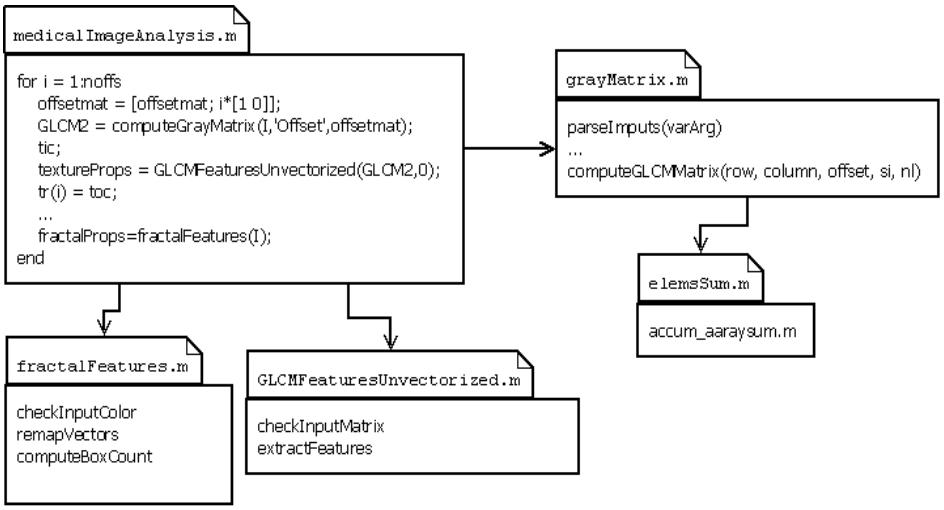

Fig. 3.3. Matlab routines integration

Once extracted, all features are written in a file. Further, the results file is used for image comparison and classification.

```
bad_cpu = isnan(v1_cpu) | isnan(v2_cpu);
bad=glogical(bad_cpu); % start using the GPU
if any(bad)
   wid = sprintf('Images:%s:are not supported',mfilename);
   warning (wid, 'Cannot compute GLCM matrix.');
end
 %if everithing okay, start processing: remove elements from the original matrix
Ind_cpu = [v1_cpu v2_cpu];
Ind=gsingle(Ind_cpu); %move the computation on the GPU
size(Ind)
if isempty(Ind)
   oneGLCM = zeros(nl);

else
   % compute the co-occurrence matrix on the GPU (using GPU-specific data) by
accumulating the elems according to their subscripts
  oneGLCM = elemSum(Ind, 1, [nl nl]);
end
```

4. Experimental results

The software system has been tested using a collection of ten test complex medical images with fractal textures; an example is provided in Fig. 4.1. The images were acquired using a high-precision dermoscope, a device which magnifies a pigmented lesion and allows the dermatologists to see and take

pictures through the bottom of the outermost layer of skin. The resolution of the test pictures starts at 1600x1200 pixels and increases until 2592x1944 pixels. When trying to execute traditional algorithms for features extraction, the runtime exceeded expectations: for the largest images, there were necessary from 20 to 150 seconds to be processed. The hardware configuration has a strong impact on both CPU and GPU execution results, thus it is imperative to consider it during performance evaluation. The presented results are based on executing the algorithms on a system configuration with an AMD Turion64 X2 TL-62, running at 2.2GHz with 2 GB DDRAM400 of memory and one CUDA-enabled GTX470 graphics card with 1280 MB GDDR5 of global memory and 448 CUDA cores.

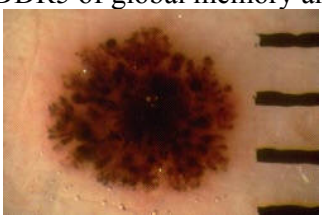

Fig. 4.1. Test image

The first implementation integrated the custom routines presented in Fig. 3.2. We have used the Matlab profiler to debug and optimize the code by tracking the execution time. We have recorded the execution time and the number of calls for each function presented in Fig. 3.2 and we discovered that using the standard *accum_arraysum.m* routine was extremely time-consuming. Therefore, we have rewritten the function in CUDA C and integrated it at runtime using Jacket SDK.

Although, Matlab is well-known as a powerful programming language, dedicated especially for complex technical computations, the code is still interpreted at runtime, whereas faster languages like C or C++ are compiled ahead of time into the computer's native language, offering significant speed improvements. Since we have chosen to implement and execute the system using Matlab, to compensate the above presented drawback, we developed a second version applying memory pre-allocation and vectorization techniques as recommended by MathWorks documentation [29]. In terms of efficiency, current results were confirmed by the results already presented in [3], i.e. the image comparison conducted to the same classification, while the processing time has been significantly reduced. The current processing time is still highly dependent on the image resolution, but, as expected, the computation speed has been increased, even if the rest of the system remained as previously presented (i.e. running on the CPU).

5. Conclusions

This paper's main objective was to present an improvement meant to continue previous work and increase the speed of texture and fractal analysis, as

used for medical images classification processes for early skin cancer detection. The system designed so far uses the GPU for the most computationally expensive operations, executing features comparison and images classification still on the CPU platform. The results both have shown that previously costly computation phases can be speed-up keeping the same results in terms of result's quality and they have also confirmed the fact that recent hardware and software technology can lead to achieving outstanding performances, especially when it comes to large amounts of data. Although, for the moment, the system has been tested only on a test database, it presents itself promising for cancer diagnosticians and, in the future, we intend to optimize it to use parallel workers for concurrent multiple images. Yet, the main downside proved to be the fact that results are highly dependent on the platform the algorithm runs on, although the software implementation is easily portable across similar hardware and the GPUs represent an inexpensive alternative to reconfigurable parallel hardware. The complex memory access patterns and the portability of the code are now almost transparent to programmer by using a runtime platform which bounds Matlab, C, or other programming languages to the graphic platform.

R E F E R E N C E S

[1] *H. Muller, N. Michous, D. Bandon, A. Geissbuhler*, "A review of content based image retrieval systems in medical applications – clinical benefits and future directions", in Instrumental Journal of Medical Informatics, **vol. 73**, no. 1, 2004, pp. 1-23

[2] *P.Y. Lau, S. Ozawa*, "An image based analysis for classifying multimodal brain images in the image-guided medical diagnosis model", in Annual International Conference of the IEEE Engineering in Medicine and Biology Society, **vol. 26**, Ed. D. Hudson, New Jersey, 2004, pp. 3400-3403

[3] *R.Dobrescu, M. Dobrescu, St. Mocanu, D. Popescu*, "Medical images classification for skin cancer diagnosis based on combined texture and fractal analysis", WSEAS Transactions on Biology and Biomedicine, **vol. 7**, no. 3, 2010, pp. 223-232

[4] *J.A. Stratton, S.S. Stone, W.W. Hwu*, "An efficient implementation of CUDA kernels for multi-core CPUs", in Proceedings of the 21st International Workshop on Languages and Compilers for Parallel Computing (LCPC), Canada, 2008

[5] *R. Dobrescu, Şt. Mocanu, Daniela Saru, A. Grumăzescu, Ramona Din,* "Aplicații pentru procesarea de imagini pe platforma CUDA – studiu de caz (CUDA Platform Based Image Processing Applications – A Case Study)", Revista Română de Informatică şi Automatică, **vol. 21**, no. 2, 2011, pp. 81-86

[6] ***, NVIDIA – GPU Computing, NVIDIA Documentation, 2012

[7] *D.B. Kirk, W.W. Hwu*, "Programming massively parallel processors. A hands-on approach", Elsevier, United States of America, 2010

[8] *Şt. Mocanu, R. Dobrescu, Daniela Saru, Ramona Din, A. Grumăzescu,* "Arhitecturi complexe folosite în prelucrarea paralelă a imaginilor (Complex Architectures Used for Image Parallel Processing)", Revista Română de Informatică şi Automatică, **vol. 20**, no. 1, 2010, pp. 97-105

[9] *J. Li, R. M. Gray, E. Y. Chang*, "Confidence-based dynamic ensemble for image annotation and semantics discovery", ACM Multimedia, 2003

[10] *P. Duygulu, K. Barnard, N. de Freitas, D. Forsyth*, „Object recognition as machine translation: learning a lexicon for a fixed image vocabulary“, in 7th European Conference on Computer Vision, 2002, pp. 97-112
[11] *I. Bartolini, P. Ciaccia, M. Patella*, "WARP: Accurate retrieval of shapes using phase of Fourier descriptors and time warping distance", IEEE Trans. Pattern Analysis and Machine Intelligence, **vol. 27**, no. 1, 2005, pp. 142-147
[12] *D. M. Blei, M. I. Jordan*, „Modeling annotated data“, in Proceedings of ACM Conference on Research and Development in Information Retrieval, 2003
[13] *K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, M.I. Jordan*, "Matching words and pictures", Journal of Machine Learning Research, **vol. 3**, 2003, pp. 1107-1135
[14] *J. Jeon, V. Lavrenko, and R. Manmatha*, "Automatic image annotation and retrieval using cross-media relevance models", in Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval, USA, 2003, pp. 119-126
[15] *R. Jin, Joyce .Y. Chai, and L. Si*, "Effective automatic image annotation via a coherent language model and active learning", in Proceedings of the 12th annual ACM international conference in Multimedia, 2004, pp. 892-899
[16] *V. Lavrenko, R. Manmatha, and J. Jeon*, "A model for learning the semantics of pictures", in Proceedings Advances in Neutral Information Processing Systems, 2003
[17] *F. Monay, D. Gatica-Perez*, "On image auto-annotation with the latent space models", IEEE Trans. Pattern Analysis and Machine Intelligence, 2003
[18] *R. Datta, J. Li, J.Z. Wang*, "Content-based image retrieval – approaches and trends of the new age", in Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, New York, USA, 2005, pp. 253-262
[19] *Yu Xiaohong, Xu Jinhua*, "The related Techniques of Content-based Image Retrieval", International Symposium on Computer Science and Computational Technology, ISCSCT '08, vol. 1, 2008, pp. 154-158
[20] *C. Vasilescu, A. Herlea, B. Ivanov, R. Dobrescu, F. Talos*, "A survey on differences between intestinal and diffuse type of gastric carcinoma", in Interdisciplinary Applications of Fractal and Chaos Theory, Romanian Academy Ed., Bucharest, 2004
[21] *A. D. Crisan*, "Image processing using fractal techniques", PhD Thesis, Politehnica University Bucharest, 2005
[22] *T. C. Wang, N.B. Karayiannis*, "Detection of microcalcifications in digital mammograms using walvets," IEEE Trans. On Medical Imaging, **vol.17**, no.4, 1998, pp. 498-509
[23] *R.M. Haralick, K. Shanmugan, I.H. Dinstein*, "Textural features for image classification", IEEE Transactions on Systems, Man and Cybernetics, **vol. 3**, 1973, pp. 610-621
[24] *A. Bouridane*, *M.A. Tahir, F. Kurugollu*, "An fpga based coprocessor for GLCM and Haralick texture features and their application in prostate cancer classification", Analog Integrated Circuits and Signal Processing, **vol. 43**, no. 2, 2005, pp. 205-215
[25] *D. A. Clausi, M. Jernigan*, "A fast method to determine cooccurence texture features using a linked list implementation", Remote Sensing of Environment, **vol. 36**, 1996, pp. 506-509
[26] *D. A. Clausi, M. Jernigan*, "A fast method to determine co-occurence texture features", IEEE Trans. On Geoscience and Remote Sensing, **vol. 36**, 1998, pp. 298-300
[27] *T. Sullivan, Heather Nelson, T. McBee, M. Alvino*, "General-Purpose Computing on Graphics Processing Units: GPU Processing of Protein Structure Comparisons", Technical Report, 2007
[28] *D. Popescu, R. Dobrescu*, *Nicoleta Angelescu*, "Statistical texture analysis of road for moving objects," U.P.B. Scientific Bulletin, Series C, **vol. 70**, 2008, pp. 75-84
[29] ***, "Techniques for improving performance," Matlab Documentation, 2012, http://www.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html#f8-790494