International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal | www.ijtsrd.com
ISSN No: 2456 - 6470 | Volume - 3 | Issue – 1 | Nov – Dec 2018

ROI Determination and Compression in MRI Using Gradient Method with CUDA

Mahmut Ünver
Department of Computer Programming, Kırıkkale University, Kırıkkale, Turkey

ABSTRACT
Due to the large use of MRI in hospitals, large storage areas are needed to store these images. Also, if you want to access these images over the system repeatedly, a large bandwidth is required. To solve this problem, it will be necessary to compress and store the medical imaging system quickly and without disruption. It has been seen that in the studies made on MRIs, the non-used regions (NON-ROI) occupy a large space and the image size can be reduced significantly when the unnecessary area in the image is cleaned. In this method developed with CUDA, the region of interest (ROI) in the MRI is detected by sending a 3x3 Kirsch filter matrix to the CUDA cores and the NON-ROI region is extracted from the image with CUDA. These operations are first executed by the serial application on CPU, then by a parallel application on GPU. As a result, the application running on the GPU produced 34 times faster results than the application on the CPU. When images are compressed with this new improved method, it takes up 89% less than the original image size and 15% less than the original compressed image.

***Key Words**: Medical Image, Parallel Programming, Medical Image Processing, CUDA, ROI.*

1. INTRODUCTION
1.1. Structure of CUDA
CPU generally remains incapable in terms of architecture to work on image processing and calculate it faster. In that case, GPU technology designed its architecture on image processing has developed. Increasing computational capabilities of GPUs have triggered the implementation of GPU work in areas such as image processing, linear algebra, statistics, and 3D modeling [1]. A parallel computing architecture called Compute Unified Device Architecture (CUDA) that increase computing performance significantly with NVIDIA's GPU (Graphics Processing Unit) capacity. Thanks to the GPU with CUDA architecture, scientists, software developers and researchers are doing research in computational biology and chemistry, image and video processing, seismic analysis and ray tracing, medical image processing. [2]. The primary version number and the secondary version number express the GPU calculation capability. Products with the same primary version numbers are in the same physical core architecture. Devices with primary version number 1 have Tesla architecture, devices 2 have Fermi architecture, devices 3 have Kepler architecture, and 5 have Maxwell architecture. The secondary version number changes as these architects evolve [3]. Tesla architecture developed in 2007 has a clear hierarchy of Tesla cards primarily in texture/processor clusters. These clusters can be scalable, can be a single, eight or more. The goal is to have a number of clusters that can be expanded easily [4]. In 2010, NVIDIA engineers started working to design a new GPU architecture. This architecture defines how a GPU is connected to each other and how they work [5]. The Kepler architecture, released by NVIDIA in 2012, offers more processing performance and efficiency with the new SM (Streaming Multiprocessor) design, which allows larger areas to be allocated to process cores instead of logic control [6]. GPUs powered by Maxwell architecture released in 2014 are GPUs that can render for the first time an indirect light dynamically using the new VXGI (Voxel Global Illumination) technology. Since the light interacts more realistically in the playing environment, the scenes are closer and more believable than the real life [7].

Medical Imaging Processing and Medical Imaging Processing with CUDA were performed in the following places where the literature search resulted:

- The work in [8, 9] is CUDA has implemented simple convolution, and FFT (Fast Fourier Transform) based convolution processes using CUFFT library and presented analyzes.

- The work in [10] is experiences on CUDA use are shared. The performance values achieved with CUDA in medical underground image processing studies are compared with the CPU implementation of the same studies.

- The work in [11] is With CUDA C; motion tracking application has been made. The practice has been tested on different GPUs, and it has been observed that even small differences between GPUs can cause significant performance differences between applications.

- The work in [12] is by applying medical image segmentation algorithms using CUDA, the performances of the GPU and CPU were measured by comparing them with other researchers, explaining the advantages of CUDA and how it was designed.

- The work in [13] is deformation image recording algorithm was applied on 3D lung tomography image with CUDA. Whatever the size of the dataset, the speed was 55 times faster than the CPU code, and the best results were obtained in this respect.

- The work in [14] is an integrated archiving method based on ROI (Interest), and OCR (Automatic Character Recognition) has been developed in medical images. In this method, the image is divided into ROI be NON-ROI regions. The ROI region is compressed with the lossless compression algorithm JPEG-LS, while the OCR and Huffman algorithms are used in the NON-ROI region. With this method, a compression ratio of 92.12% to 97.84% was obtained for the NON-ROI region of the image. In addition, this developed method has achieved an 83.30% success rate for the NON-ROI portion of the view according to the best approach in the literature.

- The work in [15] is an efficient middle-tier platform has been developed using modern technologies. Medical image data is compressed using Hadoop/Map Reduce and stored using MongoDB. In study, the developed service-based platform is available to all HIMS for medical imaging archives without changing the DICOM standard. The developed fast and efficient search engine provides access to the medical image that the end user searched for safely.

CUDA programming consists of the term Kernel, Grid, Block, and Thread.

Kernel: In CUDA programming, the part of a code that will work on the GPU is called the "kernel". The GPU creates a Kernel copy for each element of the dataset. These Kernel copies are called "thread"[16].

Grid: The Grid is the structure that the Blocks come together to create. Each Kernel call creates a Grid. Grids can be 1D, 2D or 3D. These dimensions are expressed as "gridDim.x", "gridDim.y" and "gridDim.z". "gridDim" refers to the number of blocks in the grid [16].

Block: The block structure consists of threads that have the ability to operate in parallel. They can be 1D, 2D or 3D. They are grouped in the grid. Each block has its own unique index in this 3D within the grid. These indices are "blockIdx.x", "blockIdx.y" and "blockIdx.z". The threads in it are dimensioned according to the number of rows and columns, and these dimensions are expressed as "blockDim.x", "blockDim.y", "blockDim.z". "BlockDim" means the thread size in the local block [16].

Thread: Thread is the smallest unit in the CUDA architecture. Blocks can be 1D, 2D or 3D. They run the same piece of code in parallel with each other. The threads are grouped in blocks. Threads in different blocks cannot work together [16].

1.2. Archiving of Medical Images

The archiving of medical images has an essential issue in medical documentation. Digital archiving, especially in the age of information, is more of an issue on the agenda. This is because the cost of film for printing medical images on films and the time required for archive archiving to be reused is a disadvantage of physical archiving. As image processing systems in the health field evolve, how digital images are stored together with patient information and how to access these images together with the image data is a matter of emphasis on the digital system. Because of certain standard works, the DICOM dataset standard has emerged [17].

1.3. Standard of DICOM

It was developed to find solutions to questions about how to use the archived medical image. The DICOM (Digital Imaging and Communications) file structure is the connotation of a database. As in the databases, both text data can be written in the file, and raw image data can be added. In this standard, which stores all data in a single file, there are labels to avoid confusion when retrieving the data [17].

2. DETAILS EXPERIMENTAL

2.1. Materials and Procedures

When examined in Figure1, the black area outside the region of interest (ROI), that is, the NON-ROI region, which is to be considered on the MRI, occupies a large area on the image. In this study, the ROI region used on the image will be determined by CUDA. The resulting image will be compressed and compared with the original image.

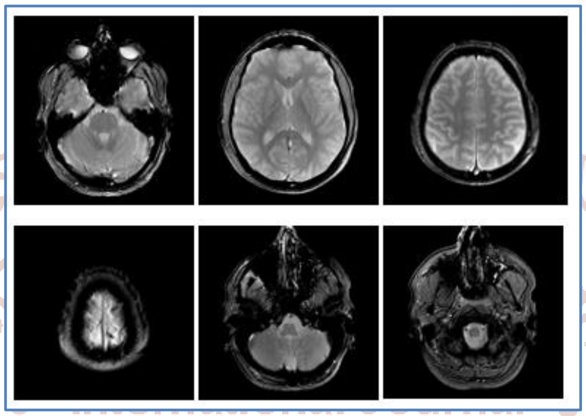
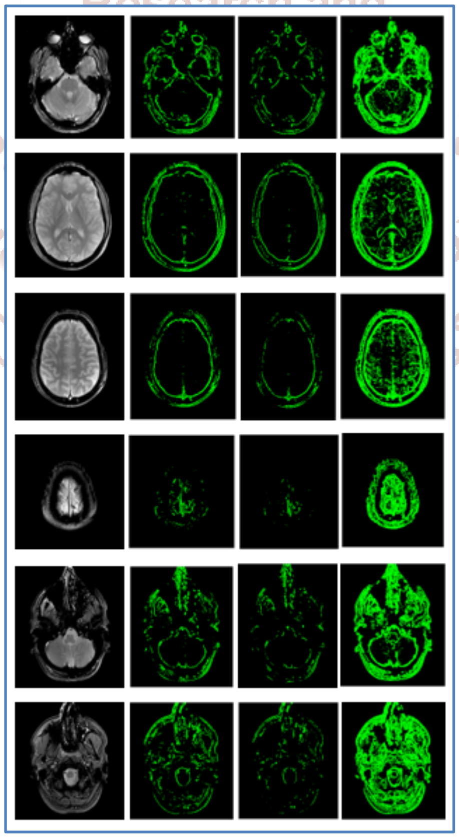


Figure 1: Brain MRI

2.2 ROI Determination



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

Figure 2: Application of different filters to brain MRI (a) original image (b) Sobel Filter Matrix (c) Prewitt Filter Matrix (d) Kirsch Filter Matrix.

When the ROI region is set on the MRI, for each pixel;

1. Gradient-based on the derivative is calculated.
   a. For best results, this process should select the most suitable Gradient matrix.
2. Calculated Gradient value is compared to the threshold value
   a. If it is greater than the specified threshold, the pixel is marked as the edge point.
   b. If not, the next pixel is passed.

When Sobel, Prewitt and Kirsch Filter Matrices are applied separately, and gradient process is applied, when the threshold value is selected by setting the threshold value 255, the Sobel and Prewitt matrices can detect the edge regions, but they are not sufficient to determine the ROI region in full sense as shown in Figure 2. The Kirsch matrix has succeeded in locating the ROI region at high resolution, even though it finds it with thick edges (expressing more than one pixel at the edge) in different types of MRI.

2.2. Extracting ROI in MRI

Once the ROI region has been determined, the extreme coordinates of the ROI region are determined by traversing the image upside down, right to left, up and down, and left to right, and the image frame is collapsed according to these coordinates.

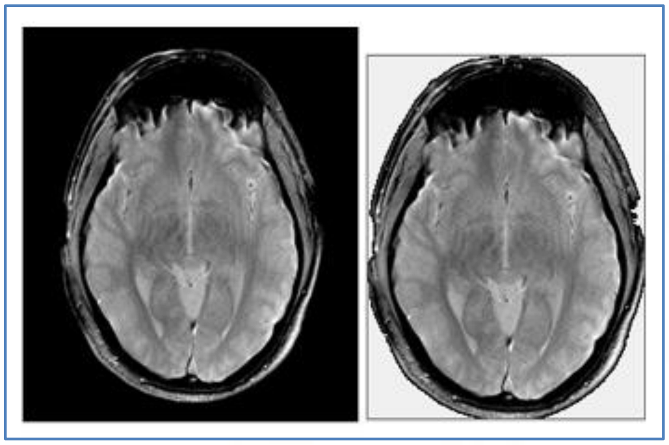

(a) (b)

Figure 3: (a) Original image (b) ROI region detected and frame-collapsed image.

3. RESULTS AND DISCUSSION

Table 1 shows the data sizes of 6 MRI shown in Figure 3.2 after the original data and the ROI image generated by the application and the mapping method after mapping are separately compressed by a compression algorithm of the resulting data. The size of the ROI image is 40% less than the original image size, and the size of the compressed ROI image is on average 89% less than the original image size. Also, the compressed data size of the ROI image is on average 15% less than the compressed size of the original image.

Table1. Compressed dimensions of original and ROI processing data

MRI Number	Original MRI Size	ROI Size	Compressed Original MRI Size	Compressed ROI Size
1	256 KB	173KB	41 KB	36KB
2	256 KB	178KB	36 KB	30KB
3	256 KB	174KB	36KB	31KB
4	256 KB	174KB	43 KB	38KB
5	256 KB	138KB	35KB	29KB
6	256 KB	79KB	21KB	17KB
Average	256 KB	153KB	35KB	30KB

Table 2 shows the runtime of the serial and parallel codes running on CPU and CUDA and GPU by mapping ROI image on 6 MRI shown in Figure 3.2 and mapping method operations afterward. The parallel GPU code written in CUDA has achieved significant success in each image according to the serial code written on the CPU and performed 34 times faster processing in average.

Table 2: Runtime comparison of CUDA-encoded GPU code with serial code running on CPU

MRI	Serial CPU Code Runtime	CUDA GPU Code Runtime
1	325 ms	9 ms
2	315 ms	10 ms
3	314 ms	12 ms
4	318 ms	9 ms
5	354 ms	13 ms
6	436 ms	8 ms
Average	343 ms	10 ms

CONCLUSIONS

With this study, the methods have been used to store MRIs with less memory space in the digital system and to achieve these images faster. It is thought that it would be more logical to hide only the ROI field because the area of the (NON-ROI) region outside the ROI of the MRI is significantly larger. From there, the ROI region was detected and removed from the image. The size of the compressed image formed as a result of this method is about 12% of the original size. An area savings of 88% for each MRI is a significant data space savings when it is thought that there are millions of MRIs stored in hospitals.

Storing the archive in the archive system is a second possibility. When the resulting image is compressed, the resulting data size is about 85% of the size of the original image when compressed. In this case, on average, an area saving of 15% is achieved for each MRI.

The application has been run on both CPU and CUDA separately on GPU and run times have been examined. According to this; the parallel application written in CUDA was seen to run 34 times faster than the application written on the CPU. This study and literature studies show that GPU systems are fast (10 - 55 times) faster than CPU performance in image processing area. The CUDA-coded parallel GPU application used in this study achieved a result above the performance averages of the GPU applications in the literature.

ACKNOWLEDGMENTS

This work is supported by the Kırıkkale University Department of Scientific Research Projects (2016/110).

REFERENCES

1. Yıldız, E., High-Performance Image Processing with NVIDIA CUDA, Master Thesis, Istanbul University Institute of Science and Technology, İstanbul, 2011.
2. CUDA, http://www.nvidia.com.tr/object/cuda-parallel-compu ting-tr.html. [Accessed 09.12.2017].
3. Sözen, N., CUDA Architecture, http://nezihesozen. github.io/mydoc/cuda6.html. [Accessed 09.12.2017].
4. Gahagan M., Tesla Architecture, https://cseweb.ucsd.edu/class es/fa12/cse141/pdf/09/GPU_Gahagan_FA12.pdf. [Accessed 10.12.2017].
5. Baskaran S., Fermi Architecture, https://www.linkedin.com/pulse/nvidia-fermi-vs-kepler-maxwell-pascal-gpus-sangeetha-baskaran. [Accessed 10.12.2017].
6. Kepler Architecture, http://www.nvidia.com.tr/object/nvidia-kepler-tr.html. [Accessed 10.12.2017].
7. Maxwell Achitecture, http://www.nvidia.com.tr/object/maxwel-gpu-architecture-tr.html. [Accessed 10.12.2017].
8. Podlozhnyuk, V., Image Convolution with CUDA, http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_64_website/projects/convolutionSeparable/doc/convolutionSeparable.pdf. [Accessed 12.12.2017].
9. Podlozhnyuk, V., FFT based 2D Convolution, http://developer.download.nvidia.com/compute/cuda/2_2/sdk/website/projects/convolutionFFT2D/doc/convolutionFFT2D.pdf, [Accessed 12.10.2017].
10. Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y. and Volkov, V., Parallel Computing Experiences with CUDA, Micro, IEEE, 28(4), pp. 13-27, 2008.
11. Huang, J., Ponce, S.P., Park, S.I, Yong, C. and Querk, F., GPU Accelerated Computation for Robust Motion Tracking Using the CUDA Framework, 5th International Conference on Visual Information Engineering, pp. 437-442, 2008.
12. Pan L., Gu L., and Jianrong X., Implementation of Medical Image Segmentation in CUDA, International Conference on Information Technology and Applications in Biomedicine, ITAB, ISBN: 978-1-4244-2254-8, 2008.
13. Owens, J., D., Ozcelik, P.M., Xia, J. and Samant, S.S., Implementation of Medical Image Segmentation in CUDA, International Conference on Computational Sciences and Its Applications, ICCSA, ISBN: 978-0-7695-3243-1, 2008.
14. Erguzen, A., Erdal, E., Medical Image Archiving System Implementation with Lossless Region of

Interest and Optical Character Recognition, Journal of Medical Imaging and Health Informatics, Volume 7, Number 6, s 1246-1252(7), 2017.

15. Erguzen, A., Erdal, E., An Efficient Middle Layer Platform for Medical Imaging Archives, Journal of Healthcare Engineering, Volume 2018,12 pages, 2018, doi:10.1155/2018/3984061.

16. Sözen, N., Grid Block Thread Structure, http://nezihesozen.github.io/mydoc/cuda2.html. [Accessed 20.12.2017].

17. Ulaş, M., Boyacı, A., Akademik Bilişim'07 - IX. Akademik Bilişim Konferansı Bildirileri, Dumlupınar University, Kütahya, pp.69-74, 2007