IOP Conf. Series: Journal of Physics: Conf. Series **803** (2017) 012152 doi:10.1088/1742-6596/803/1/012152

Digital image processing using parallel computing based on CUDA technology

I P Skirnevskiy[1], **A V Pustovit**[1], **M O Abdrashitova**[1]

[1] Tomsk Polytechnic University, 30, Lenina ave., Tomsk, 634050, Russia

E-mail: skirnevskiy@tpu.ru

Abstract. This article describes expediency of using a graphics processing unit (GPU) in big data processing in the context of digital images processing. It provides a short description of a parallel computing technology and its usage in different areas, definition of the image noise and a brief overview of some noise removal algorithms. It also describes some basic requirements that should be met by certain noise removal algorithm in the projection to computer tomography. It provides comparison of the performance with and without using GPU as well as with different percentage of using CPU and GPU.

1. Introduction

Nowadays, computers process a huge amount of data [1]. Complex mathematical algorithms are embedded in different information systems and hardware complexes in almost all areas of science and technology. It is quite often that for such algorithms as modeling of molecular dynamics [2], protein folding [3], aerodynamic processes[4], etc. a high computer performance is required. It is obvious that for such calculations the performance of common personal computers is insufficient. Moreover, if users do not work with complex computer calculations and are not interested in optimization, in science and industry, there occurs one of the main problems concerning the tendency of people to use as many resources as possible. The concept of parallel computing was invented almost at the same time as the first computer [5]. The main principles of parallel computing are described in the book of T. J. Fountain «Parallel Computing: Principles and Practice». Today, for the purpose of increasing available computing resources, clusters [6] can be used. However, their overview is beyond the scope of the study presented in the article.

Parallel computing can be implemented in central processor (CPU) [7], as well as in graphical processor (GPU) [8]. The article describes optimization of the noise removal algorithm using parallel computing on GPU. Despite the fact that the removal of digital noise in images is well-studied, one of its first references is given in article [9]. The researchers are not always focused on the algorithms performance itself as the result is much more important. There is a number of problems in which the noise removal process is a part of the overall process. In particular, the process of obtaining CT scans which diagram is shown in Figure 1. Typically, obtaining CT images is connected with processing of a huge amount of data [10], which makes the optimization one of the main questions.

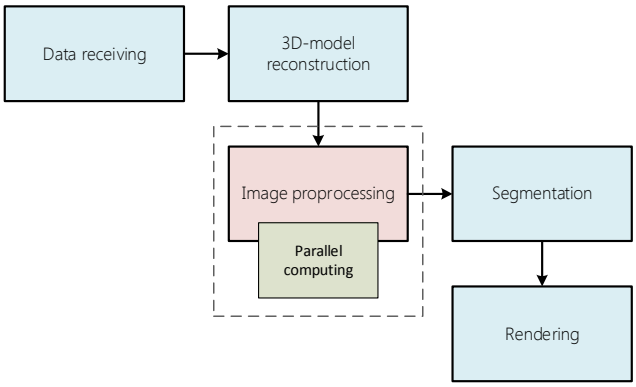


Figure 1. A computer tomography process

The relevance of the development is defined by the absence of similar free projects and works. Existing web-applications and desktop applications do not perform the tasks efficiently.

2. Relevance

The segmentation process of CT images is presented as a separation of the resulting array of data into layers (slices) and capturing a certain part in each slice [11]. To perform this work, there is a number of requirements to the original image. Algorithm requirements may differ for each segmentation, although there are a few common ones: the image should have as much contrast as possible between the background and the studied object, the image should contain digital noise not exceeding the threshold values. When the noise removal process becomes a part of the overall process, there are a few main requirements:

- minimal time;
- best result.

Thus, the article describes the efficiency of applying parallel computing in the noise removal process using a non-local means algorithm [12]. The algorithm allows obtaining the best result, although it requires a lot of calculations and, as a result, a lot of time to perform the task, which is contrary to the first requirement (minimal time). To solve this problem, the noise removal process may be distributed among the cores of central and graphical processors.

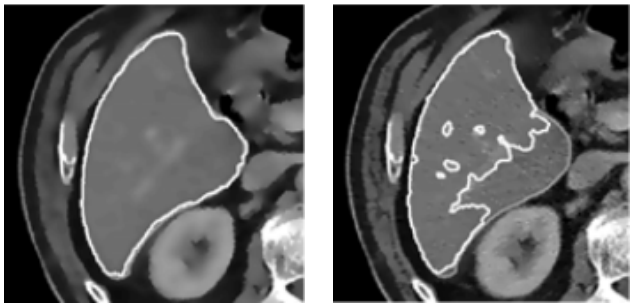
The use of parallel computing is one of the possible solutions of the problem concerning complex algorithms, as it allows using the available computing resources to the maximum as well as avoiding purchasing expensive equipment.

However, applying parallel computing for noise removal algorithms is not always advisable because of linear data or owing to the fact that it is impossible to divide the noise removal process into independent blocks of operations. The NLM algorithm scans the whole image for each pixel and detects the parts with a structure similar to a current pixel part. It is a resource intensive process. So, the obvious solution is to calculate each new pixel in its own thread.

3. Digital noise

Digital image noise is an image defect presented as random deviation of brightness or color from its original values and caused by photo sensors and device circuits [9]. Image noise can be a serious problem for future image analysis like segmentation in computed tomography.

Digital noise is a very dangerous process for segmentation as it may cause wrong object borders in images. For example, figure 2a shows a successful segmentation result with the correct object borders, figure 2b shows a wrong segmentation result [13].



a b

Figure 2. A segmentation result (2a – successful segmentation, 2b – negative segmentation)

The distorted result may be caused by the fact that the segmentation algorithm uses some noise pixels as the wrong border. Figure 2b shows that some inner pixels were recognized as the border.

Currently, there are algorithms resistant to digital noise segmentation, such as algorithm Noise-Robust Method for Image Segmentation described in [14] may even perform segmentation in the images with the noise level above the threshold values. However, such algorithms are much more complex which makes their usage inappropriate for some segmentation processes.

Successful diagnosis of human organs in medicine or detection of defects in the critical elements of industrial plants may depend on how accurate the segmentation results are.

3.1. Noise removal algorithms

All noise removal algorithms may be divided into local and non-local [3]. To calculate a new pixel value, local algorithms use the area around the pixel [3]. Meanwhile, non-local ones use pixels of the whole image, which provides a better result, though requires a lot of resources[3]. The noise removal process is a part of the preparatory step before segmentation in CT. Therefore, the appropriate choice is a non-local means algorithm. Besides, more resource intensive algorithm allows inspecting benefits of using parallel computing.

4. Parallel computing

Parallel computing is the simultaneous use of multiple computing resources in order to solve a computational problem [4]. To maximize the benefits of parallel computing, a task should consist of several independent subtasks. In the noise removal process, calculations of new pixels are independent of each other, which allows performing these calculations simultaneously.

Creation of parallel CPU threads on the GPU requires some preparation. Since GPUs do not have direct access to the computer's memory, one should first select the area of a sufficient size in the video memory, then copy the data from the random access memory (RAM). Upon completion of the calculations, it is necessary to move the data back to the RAM and clean the previously selected area of the video memory, Figure 3.

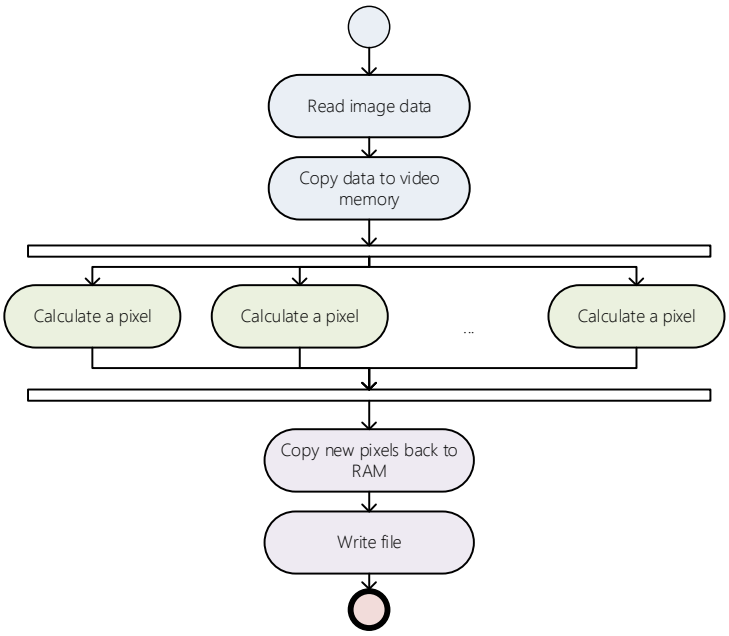


Figure 3. The GPU activity diagram

Currently, two basic technologies provide the possibility of parallel computing on the GPU. Firstly, there is CUDA technology developed by Nvidia for their own video cards. An analogue of this technology is FireStream technology from AMD. According to the research conducted in the article [15], CUDA technology makes it possible to operate the data faster, which makes this choice appropriate.

5. Parallel computing applied to a non-local algorithm

The algorithm can be applied to both monochrome and color images. The equation for calculating the new pixel values:

$$u_i(p) = \frac{1}{C(p)} \sum_{q \in \Omega} u_i(q) w(p,q), \quad (1)$$

where $u_i(p)$ – new pixel value, $u_i(q)$ – original pixel value, $w(p,q)$ – weight factor, which shows the degree of similarity of pixel areas. $C(p)$ – sum of all weights and Ω – plenty of all pixels.

$$C(p) = \sum_{q \in \Omega} w(p,q), \quad (2)$$

$$w(p,q) = e^{-\frac{max\left(d^2 - 2\sigma^2 \cdot 0.0\right)}{h^2}}, \quad (3)$$

where d – Euclidean distance between two pixels, σ – Standard deviations and h – filter parameter, usually equal to $0.4 * \sigma$

Euclidean distance can be calculated by formula

$$d^2(B(p,f), B(q,f)) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{j \in B(0,f)} (u_i(p+j) - u_i(q+j))^2, \quad (4)$$

Equation 4 is for color images, but for the monochrome image it is necessary to use Euclidean distance.

All calculations are to be done for every pixel. The obvious and the simplest solution is to perform the calculation of each pixel in a separate thread. It is worth noting that it is possible to implement a greater degree of parallelism available in a separate flow calculation of weighting coefficients (3). However, it would require thread synchronization, which makes the program code more complicated.

Using the GPU computing requires pre evaluation of the calculations. Owing to relatively simple calculations applicable to the large volume of data, it is possible that the performance of calculations on multiple cores CPU is faster than copying this volume of data to the area of video memory and back. However, one should not forget that the video memory is limited, and a large amount of data is processed in portions.

6. Experiment results

For the experiment, we will compare the performance of a non-local algorithm on an image with Gaussian noise in three cases:

- without parallel computing;
- only with CPU;
- with GPU.

There are data in table 1, which show time of the execution of the NLM algorithm with different percentage of data for CPU and GPU.

Table 1. Time of calculations

	Time (sec)	
Proportion	160x128 pixels	210x182 pixels
1 CPU thread	184.091343	627.95773
2 CPU threads	196.874450	713.950154
½ CPU + ½ GPU	98.867030	336.006175
¼ CPU + ¾ GPU	49.657433	163.791901
1/8 CPU + 7/8 GPU	39.236646	139.992048
GPU	29.888206	111.724455

The data from table 1 are shown on the chart in Figure 4

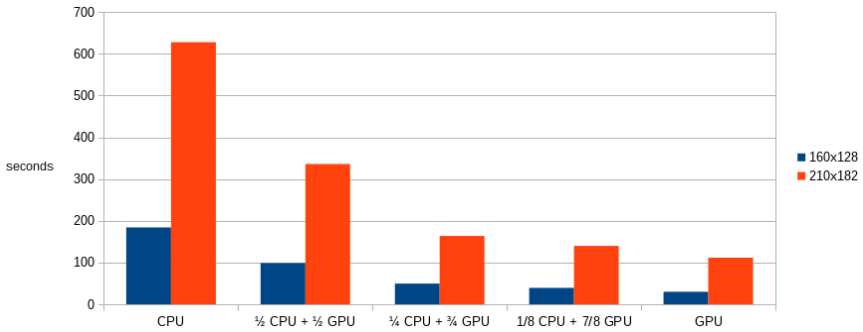


Figure 4. Time of calculation with different percentage of data for CPU dnd GPU

According to Table 1, we get the best time in the calculation only on the GPU. It should be noted that the image processing in the 2 threads on the CPU is longer than in 1 thread. This is due to the fact that creation of new threads takes time. We can also see that the productivity increases with increasing the data for GPU computing. This is due to the fact that GPU is capable of executing a lot of threads

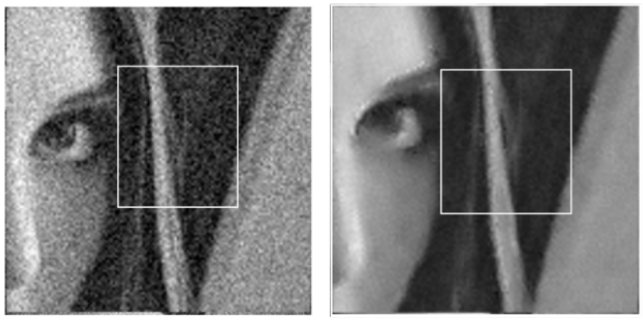
simultaneously. So, the most active threads give the most efficient performance.

Comparing the runtime without using parallel computing with the result when we transfer calculations on GPU, we obtain a 5.6 - 6-time performance boost by.

Figures 5-6 shows the NLM algorithm applied to test images.

a b

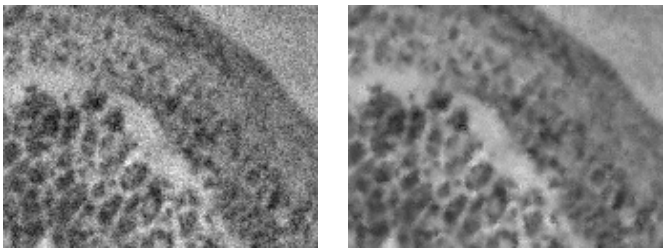
Figure 5. A test image (a – noisy image, b – filtered image)



a b

Figure 6. The part of the enlarged test image (a – noisy image, b – filtered image)

As we can see from figure 6, the NLM algorithm handles small parts of the image correctly even if they are almost the same as the background. There are a few real CT images below (figure 7)



a b

Figure 7. The example of computer tomography image processing (a – noisy image, b – filtered image)

7. Conclusion

The use of parallel computing with the GPU can significantly accelerate the implementation of the program. The degree of parallelism and the acceleration is determined by the number of independent computations performed simultaneously.

Despite the fact that the study of Nvidia CUDA technology has been presented in the field of digital image processing computer tomography, it can be used for a wide range of other applications. For example, in recent years, this technology has rapidly integrated into the program for conversion of video, data and programs. It means that it can be applicable for professional and at-home use, which indicates that the technology is widespread not only in science or industry.

In the future, it is advisable to adapt this software application for sequential processing of a series of images. Taking into consideration the results of the investigation, it is worth transferring all the calculations to the GPU, meanwhile the CPU will provide the preparation of the next image for the procession.

8. Acknowledgments

This work was in part supported by the Russian Federation Governmental Program “Nauka” N: 4.1660.2016.

References

[1] Korovin A S, Skirnevsky I P and Abdrashitova M A 2015 Web-Application For Real-Time Big Data Visualization Of Complex Physical Experiments *Proc. The 2015 Int. Siberian Conf. On Control and Communications (SIBCON) (Omsk)* vol 1 *(*Novosibirsk: IEEE Russia Siberia Section) pp 1-5

[2] Pelaia G, Vatrella A, Cuda G, Maselli R and Marsico S A 2003 Molecular mechanisms of corticosteroid actions in chronic inflammatory airway diseases *Life Sciences.* **72(14)** 1549-1561

[3] Dobson C M 2003 Protein folding and misfolding *Nature* **426(6968)** 884-890

[4] Spalart P R and Rumsey C L 2007 Effective Inflow Conditions for Turbulence Models in Aerodynamic Calculations *J. AIAA Journal* **45(10)** 2544-2553

[5] Fountain T J 2006 *Parallel Computing: Principles and Practice* (Cambridge University Press)

[6] Lin H, Kemp J and Gilbert P 2010 Computing gamma calculus on computer cluster *Int. J. of Technology Diffusion* **1(4)** 42-45

[7] Ivy K L 1988 A shared virtual memory systemfor parallel computing *Int. Conf. on Parallel Computing* pp 94-101

[8] Owens J D, Houston M, Luebke D, Green S, Stone J E, and Phillips J C 2008 GPU computing *Proc. of the IEEE* 96(5) 879-899

[9] Skirnevskij I P, Savenko I I, Pustovit A V and Capko G P Sistemy vybora metoda udaleniya cifrovogo shuma dlya razlichnyh tipov izobrazhenij 2015 *Perspektivy nauki* **6(69)** 1-9

[10] Rietzel E, Pan T, and Chen G T 2005 Four-dimensional computed tomography: image formation and clinical protocol *Medical physics* **32(4)** 874-889

[11] Rapp-Bernhardt U, Weite T, Doehring W, Kropf S and Bernhardt T M 2000 Diagnostic potential of virtual bronchoscopy: advantages in comparison with axial CT slices, MPR and mIP? *European radiology* **10(6)** 981-988

[12] Buades A, Coll B and Morel J M 2005 A non-local algorithm for image denoising. *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* vol. 2 pp 60-65

[13] Eibenberger E, Borsdorf A, Wimmer A and Hornegger J 2008 Edge-preserving denoising for segmentation in ct-images *Bildverarbeitung für die Medizin* (Springer Berlin Heidelberg) pp 257-261

[14] Despotović I, Jelača V, Vansteenkiste E and Philips W 2010 Noise-robust method for image segmentation *Proc. Int. Conf. on Advanced Concepts for Intelligent Vision Systems* (Springer Berlin Heidelberg) pp 153-162

[15] Karimi K, Dickson N G and Hamze F 2010 A performance comparison of CUDA and OpenCL *arXiv preprint arXiv:1005.2581*