Real-time imaging for Spectral Optical Coherence Tomography with massively parallel data processing

Marcin Sylwestrzak,* Maciej Szkulmowski, Daniel Szlag, and Piotr Targowski

Institute of Physics, Nicolaus Copernicus University, ul. Grudziądzka 5, PL-87100 Toruń, Poland

Received July 09, 2010; accepted September 14, 2010; published September 30, 2010

Abstract—In this paper the application of massively parallel processing of Spectral Optical Coherence Tomography (SOCT) data with the aid of a low-cost Graphic Processing Unit (GPU) is presented. The reported system may be used for real-time imaging of high resolution 2D tomograms or for presenting volume data. The overall imaging speed is over 100 frames/second for 2D tomograms built of 1024 A-scans and 9 frames/second for 3D volume images containing 100 slices of 100 A-scans. This includes t he acquisition of 2048 pixels from a CCD camera per A-scan, data transfer to the processor, all necessary processing and rendering on the screen. As a contribution, the description of a data flow and parallel processing organization in a GPU is given.

Optical Coherence Tomography is based on white light interferometry and is a non-invasive technique of imaging internal structure of objects. It has been successfully applied for *in-vivo* examination of the eye [1-2] and other objects that weakly absorb and scatter light [3]. A narrow beam of light of high spatial and low temporal coherence is directed to the object. The light backscattered and/or reflected at elements of its internal structure is collected and brought to the interference with light propagating in the reference arm of the Michelson interferometer. The signal is registered in the spectrometer and post-processed, essentially by taking a Fourier transformation. This way a single, vertical line (an A-scan) of the tomogram is recovered. To obtain a cross-sectional image (a B-scan) the examining beam is scanned over the surface of the object. By repeating this scanning procedure line by line with a shift in the direction perpendicular to the scanning direction, volume data may be collected.

One of the significant limitations of this technique is the time of data processing, usually longer than data acquisition. It limits real time volume imaging to a small amount of A-scans which decreases picture quality. Recent progress in massive parallel processing gives an interesting solution to this problem. Lately developed graphic processing units (GPUs) allow very high speed of 3D rendering, but also can execute parallel, general purpose numerical calculations with efficiency higher than the CPU. Several years ago NVIDIA® corporation presented a parallel computing architecture named CUDA™ (Compute Unified Device Architecture). For the computer game market CUDA enables creating more natural effects of fire, fluid, light *et cetera*. Simultaneously, it has been recognised as a powerful tool for solving problems, where parallel computations may be utilised, especially for molecular dynamics calculations [4], chemistry [5], quantum physics [6] and others.

The data processing in OCT is especially well suited for parallel processing. This is because in-depth information is given as a vector of data (a spectrum), independently for every point of the surface of an examined object. The numbers of such vectors (1,024 to 4,096 numbers length) vary from a few thousand for 2D imaging (a single B-scan) to few hundred thousand for high resolution 3D rendering. For every vector the same set of procedures must be applied. In the system built in our laboratory they are (in order of adoption): background subtraction, λ–k spectral remapping, numerical dispersion compensation, shaping of the spectrum to Gaussian envelope, calculating the logarithm of Fourier transformation amplitude.

Utilizing a GPU for processing OCT data has already been reported [7-9]. However, in none of these reports full processing procedure was applied. It must be emphasised that it allows imaging, but with some compromise with the quality of the final results.

In our system all steps of data processing are performed in a GPU during one flow of data. Furthermore, obtained information is not returned from the GPU to the CPU but directly displayed on the screen.

The results presented in this report were obtained with a workstation utilising Intel® Core™ i7 920 (2.67 GHz) CPU with 6 GB RAM memory and a low cost ($400) game designed graphic card: NVIDIA® GeForce® GTX 285 with 2 GB device memory. To ensure fast acquisition of spectra, a National Instruments PCIe-1429 frame grabber was employed, which allows transfer of 680 MB/s over 2 Camera Link cables. As a data source two linear cameras were tested: CCD Atmel AViiVA® SM2 capable of supplying 28,000 spectra per second (28 kHz line rate) and a very high speed CMOS camera Basler Sprint, SPL4096-140km working with the 128 kHz line rate in the 2048 pixel mode.

To collect data a laboratory-made OCT system was used (Fig. 1). As a light source a Superlum D-series Broadlighter with a central wavelength of 845nm and full

* E-mail: mars@fizyka.umk.pl

spectral width at half maximum (FWHM) of $\Delta\lambda$=107nm was employed. The setup had axial and transverse resolutions equal to 4μm and 30μm, respectively, and the imaging depth of above 2.6mm.

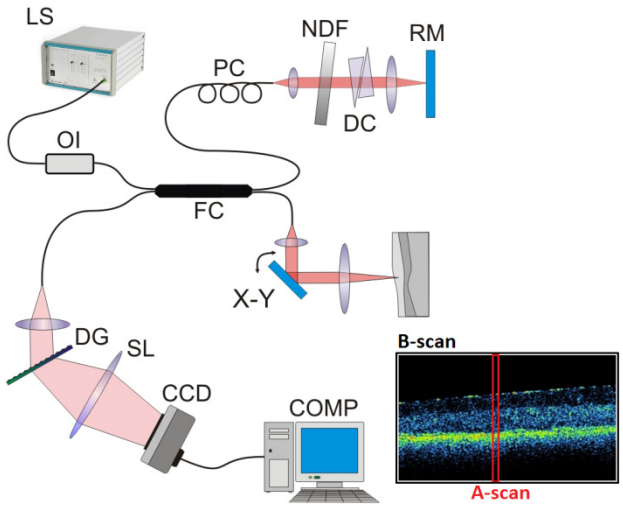

Fig. 1. OCT setup used in this study: LS – light source, OI – optical isolator, FC – fiber coupler, PC – polarization controller, NDF – neutral density filter, DC – dispersion compensator, RM – reference-arm mirror, X-Y – transversal scanner, DG – diffraction grating, SL – spectrograph lens, CCD – linear CCD camera

The software for the presented project was developed under Microsoft® Windows™ 7 Professional x64 operating system and is written in C++ programming language. The Microsoft® Visual Studio and Intel® C++ Compiler Professional Edition 11.1 with Intel® Integrated Performance Primitives library were used to ensure better efficiency of data buffering. All procedures for a GPU were prepared with NVIDIA® CUDA™ compiler version 2.3. For visualisation the OpenGL® 3.0 Library was used. To simplify some parts of code the Freeglut (an open source alternative to the GLUT – OpenGL® Utility Toolkit) library was utilised.

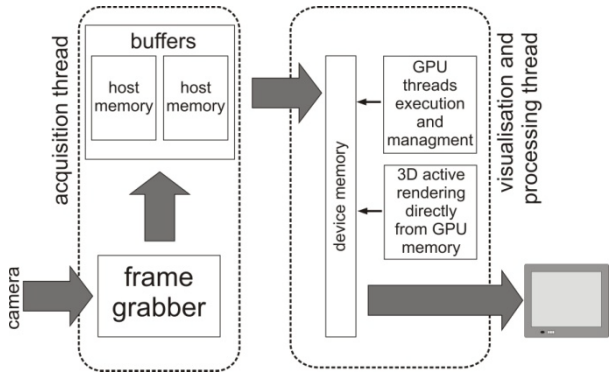

Fig. 2. Two main application threads working on the host.

The acquisition and processing software utilise two main threads running on a host computer, schematically presented in Fig. 2. The first thread is responsible for the acquisition of spectra from the camera. It is running synchronically with the OCT instrument and its main task is to copy data from the frame grabber to two memory buffers, filed alternatively. Data from each buffer represents the whole segment to be further parallel processed by a GPU. The second thread controls all GPUs and runs independently of the acquisition thread. Therefore, data is transferred to the GPU thread always when needed, e.g. when the processing of the previous set has been completed. This way acquisition and processing are not synchronised: if data collection is faster than processing, some segments will be omitted, in the opposite case the same data will be processed twice. Since every data segment contains complete information of the whole image (2D or 3D) this mode is most useful for real time imaging.

The Graphic Processor Unit utilized in this study has 30 multiprocessors comprising 8 cores each. Contrary to the main processor, GPU cores comprised in a certain multiprocessor must execute the same commands simultaneously. Every multiprocessor is equipped with its own fast shared memory. This hardware structure is used for parallel processing of a specific logical structure of treads called the Grid. It is built of Blocks, each capable of controlling the execution of 512 threads. It is essential that each Block can be executed only in one multiprocessor. Therefore, all threads of the Block have access to the fast shared memory of this multiprocessor.

As mentioned before, the elementary amount of data is in spectral OCT a single spectrum (a vector) collected from the CCD camera (Fig. 1). In the instrument used in this study this vector comprises 2048 single precision numbers. For effective parallel processing of OCT data it is vital that no information exchange is necessary between A-scans. Therefore, each A-scan can be processed by a different Block and in our case, each thread of this Block controls four data points of the A-scan. The number of Blocks is equal to the number of A-scans in the whole tomogram and, if necessary, may be very large (e.g. for 3D imaging).

Calculations performed by multiprocessors are organised in the Kernels. This is due to the technique of GPU programming: procedures executed within the graphic card (the Kernels) are programmed and compiled independently from the programming of the host. Then they may be called from the main program which results in simultaneous launch of the procedure in all treads of the GPU.

The necessity of using the sequence of Kernels results from different access to the GPU memory by procedures used for data processing (Fig. 3). Since the second step (λ-k spectral remapping) needs simultaneous access to all

spectrum points, the first Kernel utilises the Shared memory. After this, data is transferred to Registers memory separately for each thread. This memory comprises the fastest read-write registers with only one clock cycle latency. Numerical dispersion compensation and spectral shaping are performed simultaneously in threads, each controlling four data points of the spectrum. Then they are returned to the device memory and the second Kernel performs the Fourier transformation utilising CUFFT library. The obtained A-scan is processed in the third Kernel: the absolute value and the logarithm (for better visibility of image details) are performed. The data ready for displaying is returned to the GPU device memory. The last Kernel maps this memory to the screen.

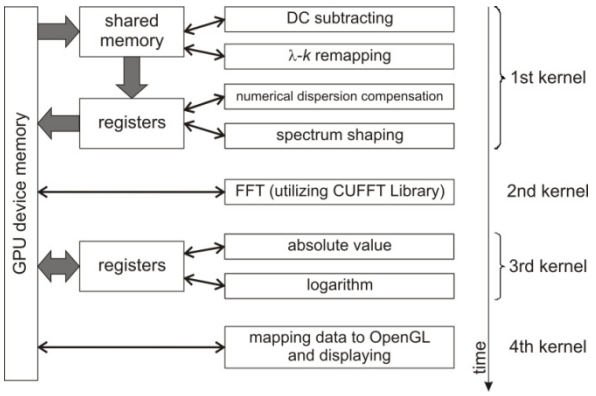

Fig. 3. GPU kernel executions, our kernels use 512 threads per A-scan.

As can be seen from Fig. 4, a significant increase of data processing speed is achieved when a large number of A-scans is parallely processed. For the system used in this study this speed stabilizes at about 300,000 A-scans/s for data bigger than 24,000 A-scans. This estimation is valid unless the size of the whole data to be processed for one image exceeds the GPU device memory capacity. For the GPU used in this study this amount is about 110,000 A-scans (330×330 volume image).

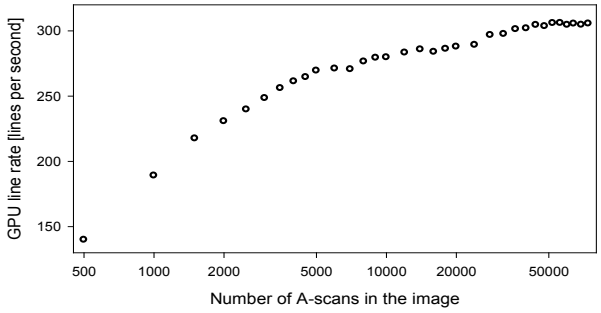

Fig. 4. GPU line rate as a function of the number of A-scans.

For 3D imaging, especially written in OpenGL procedures are utilised [10]. This approach takes advantage of very fast communication between GPU device memory and screen. By dint of this technology, fast, real–time imaging is possible.

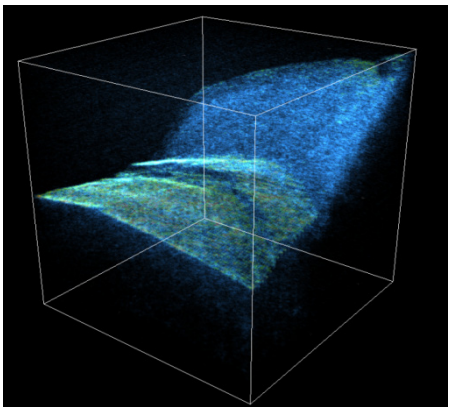

Fig. 5. An example of 3D, real time image comprising 100×100 A-scans. A finger nail with a cuticle is shown. Cage dimensions (W, D, H): 3.5x3.5x2.7 mm.

The overall rendering speed of the image plane built of 1024×1024 pixels is over 100 frames/sec in the case of 2D imaging (1024 A-scans build of 1024 pixels each). In the case of 3D imaging (Fig. 5) nine perspective pictures per second are generated. Every picture comprises 100 planes (B-scans) built of 100 A-scans each. As it can be seen from the figure this simplified image is sufficient for real time inspection of the object. Contrary to previous reports, the results presented in this contribution were obtained after full processing, and thus without compromising image quality. It is important for on-fly image evaluation in case of further recording and post-processing.

This project is funded by Polish Government Research Grant through years 2008–2011 and 7th FP *CHARISMA* project. M.Syl. and D.Sz. acknowledge support from the "Step into the Future III" program co-financed by the European Social Fund and Polish Government. M. Sz. is supported by the Foundation for Polish Science (scholarship START 2010).

References

[1] M. Wojtkowski *et al.*, Am. J. Ophthalmol. **138**, 412 (2004).
[2] M. Wojtkowski *et al.*, Ophthalmology **112**, 1734 (2005).
[3] D. Stifter, Applied Physics B Lasers and Optics **88**, 337 (2007).
[4] J.E. Stone *et al.*, Journal of Computational Chemistry **28**, 2618 (2007).
[5] I.S. Ufimtsev, T.J. Martinez, J. Chem. Theory Comput. **4**, 222 (2008).
[6] E. Gutierrez *et al.*, Computational Science **5101**, 700 (2008).
[7] J. Probst, P. Koch, G. Hüttmann, Proc. SPIE **7372**, 7372_0Q, (2009).
[8] S. Van der Jeught *et al.*, JBO Letters **15**, 30511 (2010).
[9] K. Zhang, J.U. Kang, Opt. Exp. **18**, 11772 (2010), http://www.opticsinfobase.org/oe/abstract.cfm?uri=oe-18-11-11772
[10] M. Sylwestrzak *et al.*, Proc. SPIE **7391**, 73910A (2009), http://www.fizyka.umk.pl/~ptarg/abstrakty/Application of GOP Sylwestrzak.pdf

