Real-time resampling in Fourier domain optical coherence tomography using a graphics processing unit

Sam Van der Jeught, Adrian Bradu, and Adrian Gh. Podoleanu
University of Kent, School of Physical Sciences, Applied Optics Group, CT2 7NH, Canterbury, United Kingdom

Abstract. Fourier domain optical coherence tomography (FD-OCT) requires either a linear-in-wavenumber spectrometer or a computationally heavy software algorithm to recalibrate the acquired optical signal from wavelength to wavenumber. The first method is sensitive to the position of the prism in the spectrometer, while the second method drastically slows down the system speed when it is implemented on a serially oriented central processing unit. We implement the full resampling process on a commercial graphics processing unit (GPU), distributing the necessary calculations to many stream processors that operate in parallel. A comparison between several recalibration methods is made in terms of performance and image quality. The GPU is also used to accelerate the fast Fourier transform (FFT) and to remove the background noise, thereby achieving full GPU-based signal processing without the need for extra resampling hardware. A display rate of 25 frames/sec is achieved for processed images (1024×1024 pixels) using a line-scan charge-coupled device (CCD) camera operating at 25.6 kHz. [DOI: 10.1117/1.3437078]

Keywords: Fourier domain optical coherence tomography; graphics processing unit.

Paper 10068LRR received Feb. 10, 2010; revised manuscript received Apr. 29, 2010; accepted for publication Apr. 30, 2010; published online Jun. 3, 2010.

Optical coherence tomography (OCT) is a noncontact, noninvasive interferometric technique allowing micrometer resolution imaging of tissues. OCT can be classified into two categories: time domain (TD) OCT and spectral domain (SD) OCT. SD-OCT eliminates the need for mechanical scanning in TD-OCT to produce reflectivity profiles (A-scans) by recording the interference signal as a function of wavelength. This can be done either by using a spectrometer when performing Fourier domain (FD)-OCT, to spatially encode the different optical frequencies, or by encoding them in time using a frequency scanning source, when performing swept source OCT. Both SD-OCT schemes have advantages of a better signal-to-noise ratio than that of the TD-OCT and a significantly improved imaging speed.[1] The increased imaging speed requires very fast digital signal processing that is able to keep up with over tens of megahertz video rate data acquisitions. OCT digital signal processing mechanisms based on current central processing units (CPUs) are not capable of coping with such data processing speeds, and limit the sample rate dramatically.

The rapid development of graphics processing units (GPUs) has introduced a revolution in numerical calculations. A GPU consists of many stream processors acting concurrently, thus favoring parallel programming. A recent report describes a GPU-based FD-OCT setup, where signal processing was performed partly by optical hardware and partly by the GPU[2] to enable real-time OCT imaging. We show here that similar data rates are possible by transferring all processing tasks to the GPU.

FD-OCT employs a spectrometer to disperse the output of the interferometer onto a linear digital camera. A fast Fourier transform (FFT) of the linear camera signal provides the A-scan. However, the FFT operates in the optical frequency domain, while the equally spaced camera pixels produce an approximately linear representation of the spectrum in wavelength, not in frequency. Unless linearization of data is performed in equidistant frequency slots, the peaks in the A-scan suffer shape distortions and attenuation.

The process of linearization requires data resampling and is the most computationally demanding task in generating the FD-OCT image. To avoid these time-consuming resampling calculations, an optics hardware solution was proposed[3] to achieve real-time imaging, in which the linearization of the interference signal was implemented using a linear-in-wavenumber spectrometer. This method places a customized prism behind the diffraction grating to evenly distribute the spectrum in optical frequency or optical wavenumber k. This distribution is only approximately linear and is very sensitive to the alignment of the prism. Alternatively, the use of a custom-built field programmable gate array (FPGA) integrated circuit to hard-wire the calibration algorithm was reported.[4] This hardware is expensive and the customization requires specific expertise in this technology. In this work we propose a GPU-based resampling solution that is capable of following the rate of data acquisition. In addition, the other processing steps required in FD-OCT imaging such as computing the FFT and subtracting background noise are also implemented on the GPU. This enables fully processed FD-OCT images to be displayed in real time without the need for any resampling hardware. Such a solution is superior to previously mentioned hardware-based methods, as it results in a more compact, stable, and flexible spectrometer that can be easily recalibrated when the optical source or diffraction grating is changed.

The customized GPU program was written in NVIDIA's (Santa Clara, California) dedicated software environment compute unified device architecture (CUDA),[5] compiled using Microsoft Visual Studio 2008 and tested with a commercial Geforce 9800GT on a PC with eight Intel Xeon E5405 processing cores. The graphical user interface was created in Labview 2009.

A schematic of the employed FD-OCT imaging system is illustrated in Fig. 1. Light coming from the source [SLD Broadlighter S840, Superlum (County Cork, Ireland) output power of 20 mW, centre wavelength at 840 nm, full spectral

Address all correspondence to: Sam Van de Jeught, E-mail: sv73@kent.ac.uk

1083-3668/2010/15(3)/030511/3/$25.00 © 2010 SPIE

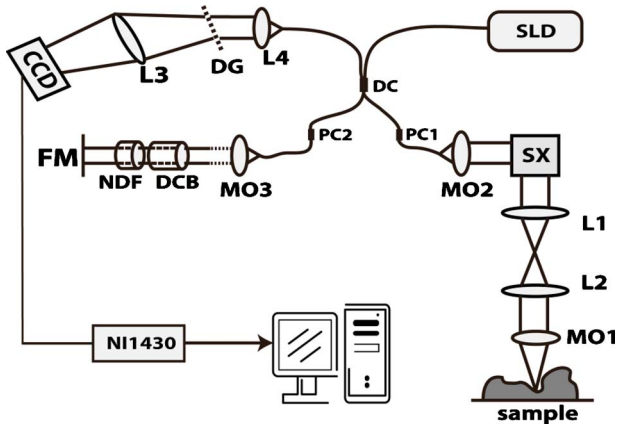


Fig. 1 Schematic of the FD-OCT imaging system. SLD, superluminescent diode; MO1-4, microscope objectives; L1 to L3, achromatic lenses; DC, directional coupler; PC1-2, polarization controllers; SX, scanning mirror; DG, transmissive diffraction grating; FM, flat mirror; DCB, dispersion compensation block; NDF, neutral density filter; NI1430, National Instruments acquisition board.

width at half maximum of 45 nm] is split by the 10/90 directional coupler into the sample arm and the reference arm. The interference of backscattered light returning from both arms is measured by a spectrometer equipped with a silicon linear charge coupled device (CCD) camera [Atmel (San Jose, California) AVIIVA M2, 2048 pixels, each 14 μm wide]. In our custom spectrometer design we used a 75-mm focal length achromatic collimating lens (MO3), a 1200 lines/mm transmission diffraction grating (Wasatch Photonics, Logan, Utah), and a 150-mm focal length achromatic lens (L3). Data from the linear CCD was transferred to a PC via a Cameralink cable and a high speed PCI frame grabber [National Instruments (Austin, Texas) NI-1430]. Residual dispersion of the system caused by MO1, L1, L2, and the optical fiber mismatch has been compensated for by using a dispersion compensating block (DCB) containing a pair of lenses L1 and L2, and a 5-mm-thick BK7 slide.

An interference image (1024 axial pixels $\times$ 1024 transversal pixels) is captured in host memory and is transferred to the GPU memory, type converting every pixel value from a 16-bit integer to a 32-bit float. Current GPUs are provided with texture memory, which is usually implemented as specialized RAM. This digital storage type is designed to accelerate frequently performed graphical operations such as the mapping of 2-D skins onto 3-D polygonal models by empowering the texture elements or *texels* with a hard-wired linear and nearest-neighbor interpolation mechanism. The image pixels are stored in a *CUDA array* and are bound to a designated region in this texture memory. By interpolating between the image texels at predefined calibration values, a resampled interference spectrum is created where the desired *k*-values are evenly distributed. After this resampling process, a prerecorded ensemble average of spectra was deducted from each A-scan to remove the noise induced by the CCD and to remove the autocorrelation signal coming from the reference arm.[6] Then, the Fourier transform of each A-scan was calculated using NVIDIA's dedicated CUDA FFT library (CUFFT). The return from complex Fourier space to real space was made by taking the absolute value or modulus of each complex pixel value. Finally, the data type is converted from a 32-bit float to a 16-bit integer and is sent back to the host memory. The signal processing flowchart of our real-time GPU-based FD-OCT imaging system is illustrated in Fig. 2.

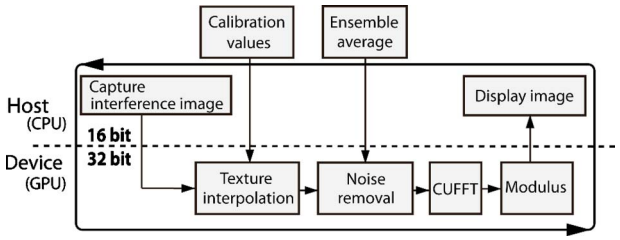
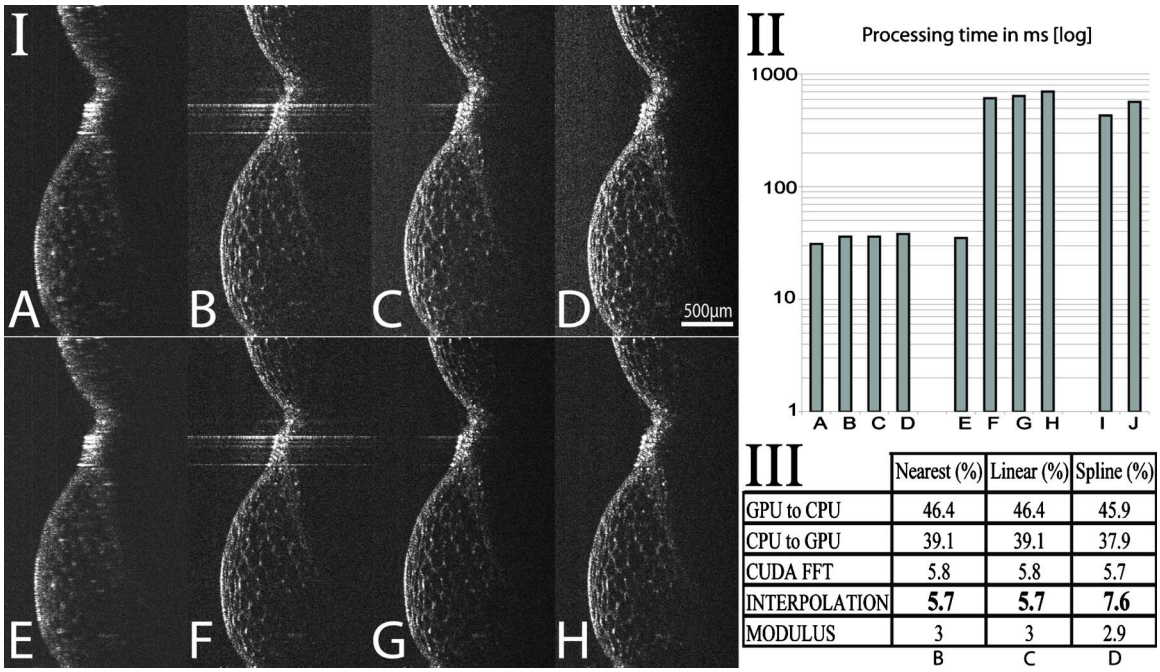


Fig. 2 Signal processing flowchart of the real-time FD-OCT imaging system.

The resampling algorithm employed in the λ to *k* conversion is crucial for the performance and imaging quality of the FD-OCT setup, as it determines the level of accuracy that is achieved in computing the desired *k*-values. Three popular interpolation methods were implemented on the GPU: zero-order (nearest neighbor), first order (linear), and third order (cubic) B-spline interpolation. Using CUDA, the first two methods are hard-wired through texture memory. Recently, efficient GPU-based cubic B-spline interpolation was achieved[7] by replacing the usage of look-up tables by a set of linear texture interpolation calculations. The resulting FD-OCT processing algorithms were benchmarked and compared to both native Labview 2009 and optimized CPU-based codes. High performance computing was achieved by the GNU Scientific Library (GSL),[8] which consists of highly optimized mathematical functions.

In part 1 of Fig. 3, a comparison in image quality is made between FD-OCT images of an orange pulp that have been resampled using different interpolation algorithms. The images in the top row (A to D) illustrate the result of increasingly accurate GPU-based interpolation. Starting with image A that was processed without any resampling, image B was resampled using nearest-neighbor interpolation, image C using linear interpolation, and image D using cubic spline interpolation. The images in the bottom row (E to H) were processed in Labview by corresponding interpolation methods on the CPU. In terms of accuracy, there is clearly an improvement in image quality with higher orders of B-spline interpolation. However, no qualitative difference between the corresponding CPU- and GPU-processed images can be noticed.

Part 2 of Fig. 3 displays the processing times for fully processed FD-OCT imaging cycles for each of the different resampling algorithms. These time measurements were made in Labview using its built-in profiler function. Note that the time required for nearest-neighbor (B) and linear (C) interpolation are exactly the same due to the hard-wired texture memory mechanism that reduces value evaluation to a single memory readout. Measurements I and J represent the GSL linear and cubic spline interpolation methods, respectively, each accompanied by the GSL FFT algorithm. The CUDA- and GSL-based algorithms were implemented in Labview through dynamic link libraries (DLLs), and all CPU processing algorithms were optimized to be executed on one CPU core. For all GPU-based interpolation methods, one can notice a speed-up of 15 to 20 times when compared to native Lab-



	Nearest (%)	Linear (%)	Spline (%)
GPU to CPU	46.4	46.4	45.9
CPU to GPU	39.1	39.1	37.9
CUDA FFT	5.8	5.8	5.7
INTERPOLATION	**5.7**	**5.7**	**7.6**
MODULUS	3	3	2.9
	B	C	D

Fig. 3 Part 1: comparison in FD-OCT B-scan image quality between resampling methods that use different interpolation algorithms. The B-scan OCT images in the top row were processed entirely on the GPU. The images in the bottom row were processed on the CPU. A, E: no calibration; B, F: nearest-neighbor interpolation; C, G: linear interpolation; D, H: cubic spline interpolation. The scale bar represents 500 μm. Part 2: corresponding processing times for the full FD-OCT processing cycles. Measurements I and J represent CPU-processed FD-OCT linear and spline interpolation algorithms, respectively, optimized using GSL. Images associated with I and J are indistinguishable from G and H, respectively. Part 3: table of relative processing times of the individual GPU-based processing cycles based on nearest neighbor (B), linear (C), and cubic spline (D) resampling. These cycles include the transfers from and to GPU memory (GPU to CPU and CPU to GPU), the GPU-based Fourier transform (CUDA FFT), the interpolation calculations (INTERPOLATION), and the return from complex to real space (MODULUS).

view algorithms, and a speed-up of 10 to 15 times when compared to the GSL algorithms. Although a low-cost graphics card was used, processing times under 40 ms were achieved for all of the employed interpolation methods, enabling real-time 25-fps FD-OCT imaging.

The table in part 3 of Fig. 3 represents the relative processing times of the individual GPU-based processing algorithms corresponding to nearest neighbor (B), linear (C), and cubic spline (D) interpolation, and was benchmarked using the NVIDIA CUDA profiler. As expected, the two timing profiles of B and C completely coincide. We can also notice that about 85% of the total processing time in all three GPU resampling methods is taken up by transferring the data to and from the GPU memory. Therefore, the isolated cubic spline (D) interpolation function (INTERPOLATION) takes up only $\sim$2% more of the total processing time than its lower order interpolation counterparts, enabling higher order interpolation accuracy without losing displaying speed.

In conclusion, a more flexible, software-based resampling scheme was proposed by offloading the λ to k conversion and all other FD-OCT processing tasks to a commercial GPU. This has enabled real-time video rate (25 Hz) processing and displaying of OCT images of size 1024×1024 pixels using a low-cost graphics card.

Acknowledgments

Van der Jeught acknowledges the Marie Curie training site, MEST-CT-2005-020353, supported by the European Commission. Bradu acknowledges the support of EPSRC grant EP/H004963/1.

References

1. R. Leitgeb, C. Hitzenberger, and A. Fercher, "Performance of Fourier domain vs. time domain optical coherence tomography," *Opt. Express* **11**, 889–894 (2003).
2. Y. Watanabe and T. Itagaki, "Real-time display on Fourier domain optical coherence tomography system using a graphics processing unit," *J. Biomed. Opt.* **14**, 060506 (2009).
3. V. M. Gelikonov, G. V. Gelikonov, and P. A. Shilyagin, "Linear-wavenumber spectrometer for high-speed spectral-domain optical coherence tomography," *Opt. Spectrosc.* **106**, 459–465 (2009).
4. A. W. Schaefer, J. J. Reynolds, D. L. Marks, and S. A. Boppart, "Real-time digital signal processing-based optical coherence tomography and Doppler optical coherence tomography," *IEEE Trans. Biomed. Eng.* **51**, 186–190 (2004).
5. Nvidia CUDA Zone, see http://www.nvidia.com/object/cuda_home.html.
6. R. K. Wang and Z. H. Ma, "A practical approach to eliminate autocorrelation artefacts for volume-rate spectral domain optical coherence tomography," *Phys. Med. Biol.* **51**, 3231–3239 (2006).
7. D. Ruijters, B. M. ter Haar Romeny, and P. Suetens, "Accuracy of GPU-based B-spline evaluation," *Proc. 10th IASTED Intl. Conf. Computer Graphics Imaging (CGIM)*, pp. 117–122 (2008).
8. See http://www.gnu.org/software/gsl/.