Journal of Innovative Optical Health Sciences
Vol. 4, No. 1 (2011) 89−95

DOI: 10.1142/S1793545811001277

TIME-DOMAIN INTERPOLATION ON GRAPHICS PROCESSING UNIT

XIQI LI[*,†,‡], GUOHUA SHI[*,†,‡,§] and YUDONG ZHANG[*,†]
[*]*The Laboratory on Adaptive Optics*
Institute of Optics and Electronics
Chinese Academy of Sciences
Chengdu 610209, China
[†]*The Key Laboratory on Adaptive Optics*
Chinese Academy of Sciences
Chengdu 610209, China
[‡]*Graduate School of the Chinese Academy of Sciences*
Beijing 100039, China
[§]*guohua_shi@yahoo.com.cn*

The signal processing speed of spectral domain optical coherence tomography (SD-OCT) has become a bottleneck in a lot of medical applications. Recently, a time-domain interpolation method was proposed. This method can get better signal-to-noise ratio (SNR) but much-reduced signal processing time in SD-OCT data processing as compared with the commonly used zero-padding interpolation method. Additionally, the resampled data can be obtained by a few data and coefficients in the cutoff window. Thus, a lot of interpolations can be performed simultaneously. So, this interpolation method is suitable for parallel computing. By using graphics processing unit (GPU) and the compute unified device architecture (CUDA) program model, time-domain interpolation can be accelerated significantly. The computing capability can be achieved more than 250,000 A-lines, 200,000 A-lines, and 160,000 A-lines in a second for 2,048 pixel OCT when the cutoff length is $L = 11$, $L = 21$, and $L = 31$, respectively. A frame SD-OCT data (400A-lines $\times$ 2,048 pixel per line) is acquired and processed on GPU in real time. The results show that signal processing time of SD-OCT can be finished in 6.223 ms when the cutoff length $L = 21$, which is much faster than that on central processing unit (CPU). Real-time signal processing of acquired data can be realized.

Keywords: Optical coherence tomography; real-time signal processing; graphics processing unit; GPU; CUDA.

1. Introduction

Optical coherence tomography (OCT) is a non-invasive, cross-sectional optical imaging technique that allows for high-sensitivity, high-resolution imaging of backscatter samples.[1−3] Due to the high sensitivity and high imaging speed, spectral domain optical coherence tomography (SD-OCT) has undergone rapid development. SD-OCT is based on spectral interferometer and detects the interference by a line scan charge coupled device (CCD). The interference signal $I(k)$ can be expressed as

$$I(k) = S(k)\left(1 + 2\int_0^\infty a(z)\cos(2knz)dz + \int_0^\infty\int_0^\infty a(z)a(z')e^{-i2kn(z-z')}dzdz'\right), \quad (1)$$

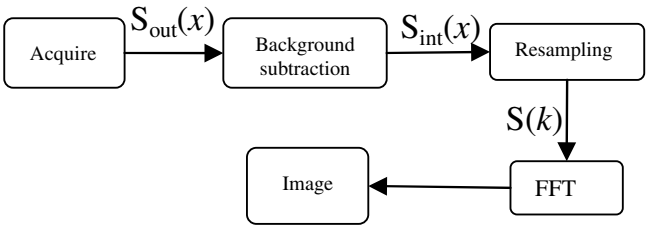

Fig. 1. Flowchart of SD-OCT signal processing procedure.

where $I(k)$ denotes the detected spectrum; $S(k)$ denotes the spectral intensity distribution of light source; $a(z)$ is the scattering amplitude. The $a(z)$ can be obtained by Fourier transformation of the detected spectrum $I(k)$. It is obvious that the intensity information is coded in the wavenumber k. Unfortunately, the CCD in SD-OCT detects the signal evenly in wavelength λ. It needs resampling the data from wavelength space to wavenumber space, or else, it leads to coherence envelop broadening and fall-off degradation in SD-OCT systems.[4] Figure 1 shows the signal processing procedure. The intensity data $S_{\text{out}}(x)$ of the spectrometer is acquired by the linescan CCD. The interference spectrum $S_{\text{int}}(x)$ is obtained after reducing the dc terms and the low-frequency noises by the background subtraction procedure. The resampled data $S(k)$ is obtained by a nonuniform sample, which is distributed linearly in wavenumber space.

Resampling is a computational complexity part for the signal processing of SD-OCT system. Because k and λ has a relationship of $k = 2\pi/\lambda$ and $\Delta k = 2\pi\Delta\lambda/\lambda^2$, and the frequency oscillation period is as small as 2 pixels in the spectrometer, any resulting local interpolation scheme (such as cubic spline) is bound to fail.[4] Zero-padding interpolation together with linear interpolation is commonly used to resample in SD-OCT imaging processing. The zero-padding interpolation needs huge fast Fourier transform algorithm (FFT) operations, which are computationally expensive. Thus, a lot of SD-OCT systems are postprocessed. Real-time optical coherence tomography processing by using digital signal processing (DSP) or field-programmable-gate-array-based (FPGA) was proposed,[5–8] which increases the system complexity and cost as additional hardware inclusions are needed. A real-time swept source polarization-sensitive OCT system by using OpenMP model has been reported,[9] but that needs expensive multicore central processing unit (CPU).

In the last 10 years, graphics processing unit (GPU) becomes more and more powerful in parallel computing.[10] Its special architecture is well suitable for the problems that can be parallely executed. Commodity GPU like NVIDIA's GTX 295 has 480 processing cores and can achieve 1784 G floating octal point (FLOP) of computational horsepower, which is much faster than that of the current CPU. But traditional tools for computing on GPU were too complex to program, limiting the nongraphical use of GPU. Recently, NVIDIA Corporation develops a tool for computing on GPU, which is named as the compute unified device architecture (CUDA) programming model.[10] With the help of CUDA model, a general purpose computing application can be developed easily. A real-time 4D signal processing and visualization using GPU on a regular nonlinear-k Fourier-domain OCT system is reported.[11] But the linear spline interpolation (LSI) they used can barely improve the signal-to-noise ratio (SNR) of the SD-OCT system. A high-speed and high-accuracy interpolation method is needed.

A time-domain interpolation for Fourier domain optical coherence tomography was reported.[1] This method not only obtains a better SNR but also gets a faster computing speed than commonly used zero-padding interpolation.[1] Furthermore, by this interpolation method, the linear k space data can be obtained by convoluting the original data detected in CCD with interpolation coefficients.[1] Thus, only the data and coefficients in cutoff window are processed. So, this interpolation method is suitable for parallel computing.

This paper combines the time-domain interpolation with the real-time computing capability of the GPU to accelerate the processing speed of SD-OCT signal processing. The time-domain interpolation processing capability can be achieved 250,000 A-lines, 200,000 A-lines, and 160,000 A-lines in a second for 2,048 pixel OCT when the cutoff length is $L = 11$, $L = 21$, and $L = 31$, respectively. A program is also developed to realize real-time signal processing of SD-OCT.

2. Time-Domain Interpolation

In SD-OCT system, interference signal $x_1(n)$ is acquired by the pixels of the linescan CCD, whose corresponding wavelengths are linearly distributed in wavelength space. We assumed that there is a sequence of wavelengths linearly distributed in k space, whose interference data are $x_2(s)$. The virtual position index s is obtained by mapping this sequence

to CCD pixel index n. The interpolated data can be obtained by using the following equation[1]:

$$x_2(s) = \frac{1}{N+1} \times \sum_{n=0}^{N-1} x_1(n)\left\{1 + 2\sum_{j=1}^{\frac{N}{2}} \cos\frac{2\pi}{N} j(s-n)\right\}. \quad (2)$$

The range of index n and s is determined by the pixels of the CCD N and the resolution of the OCT system. The coefficients are defined as

$$C(\Delta) = \frac{1}{N+1}\left(1 + 2\sum_{j=1}^{\frac{N}{2}} \cos\left(\frac{2\pi}{N} j\Delta\right)\right), \quad (3)$$

where $\Delta = s - n$. Equation (3) shows that the coefficients do not relate to the data. The interpolation can be processed if s and n are known. The virtual CCD position of the linear k space can be obtained by calibrating the spectrometer of the SD-OCT system.

It has been reported that only a few number ranges of $C(\Delta)$ contribute to the interpolation. Therefore, a narrow window can be used to cut off the $C(\Delta)$ with an acceptable accuracy to reduce the computation time, whose cutoff width is L.[1] The larger L can give a better accuracy, but it would consume more computing time. Thus, the cutoff width L is determined by a tradeoff between the accuracy and the computation time.[1] The interpolation can be finally expressed as

$$x_2(s) = \sum_{n=Min}^{Max} x_n H_{N\times N}(n, s_n) W(n), \quad (4)$$

where Min and Max are determined by the cutoff length and the position s. $H_{N\times N}$ is the coefficient matrix. W is the cutoff window. A rectangle window is suitable. Equation (4) shows that $x_2(s)$ only relates to several original data and the coefficients range in the cutoff window. So, the sequence $x_2(s)$ can be processed by a lot of parallel streams.

3. Signal Processing on GPU

A high-sensitivity and high-signal processing speed SD-OCT system is developed. Figure 2 shows the setup. The light source is a SLD (Superlum, Russia, SLD-371-HP1) with a bandwidth of 45 nm FWHM at 840 nm. The spectrometer consists of a linescan camera (AVVIVA, SM2 CL2015, 2,048 pixels), a transmission grating [Wasatch Photonics, 1200 (lines × pairs)/mm at 830 nm], and an achromatic lens ($f = 150$ mm). Two galvo mirrors are driven by a waveform generator (PCI 6221, National Instrument) to control the scan light. The interference is detected by the CCD and transmitted to computer after it is grabbed by the image acquisition device (IMAQ) (PCIe 1430, National Instrument). A computer with a quad-core CPU Intel Quad-core Q9600 at 2.6 GHz and a GPU NVIDIA GTX295 is used as the data acquisition and processing computer. The operating system was Microsoft Windows XP pro SP2 and the software developing platform was Microsoft Visual Studio 2005 SP1.

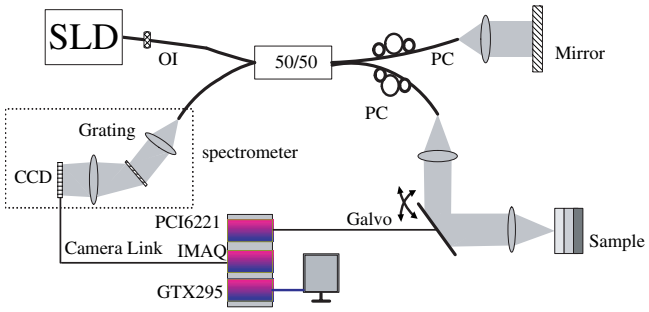

Fig. 2. The setup of a SD-OCT system. SLD, superluminescent diode; PC, polarization controller; OI, optical isolator; CCD, charge coupled device.

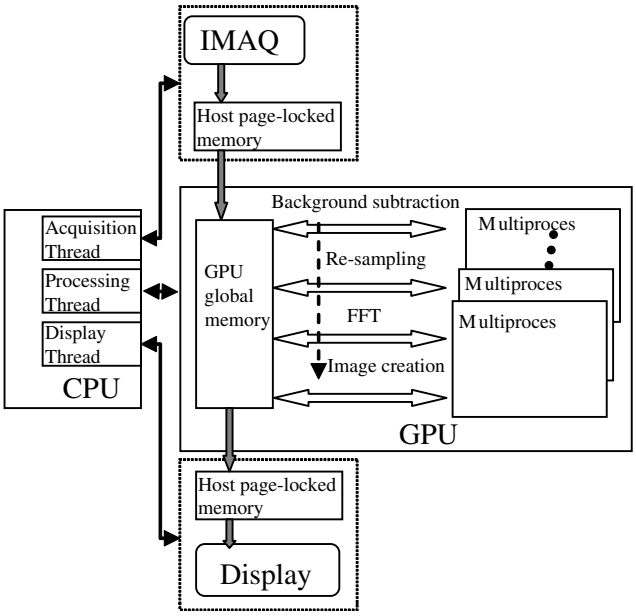

Fig. 3. The SD-OCT system's software architecture.

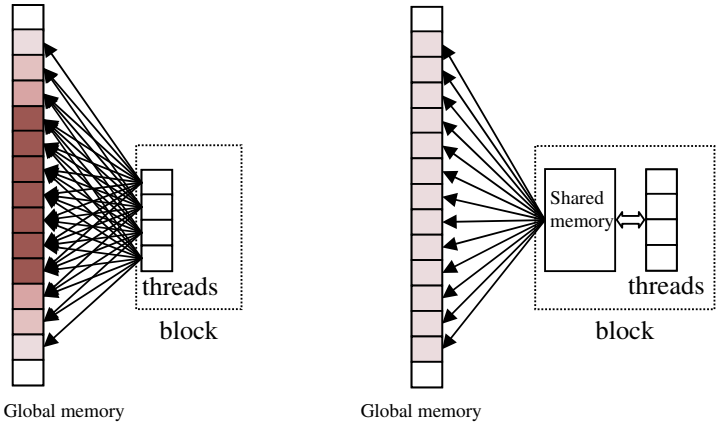

Fig. 4. Bank conflicts in time-domain interpolation. (a) Without shared memory. (b) With shared memory.

Figure 3 shows the software architecture of the SD-OCT system. The spectra data is acquired by IMAQ and transmitted to a bulk page-locked memory, in which the system obtains a higher transmission performance. And then, the data is transmitted to the global memory of the GPU for parallel computing. The spectra data is processed on GPU by four steps: background subtraction, resampling, FFT, and image creation. These steps are processed step-by-step, but every step processes the data by a lot of threads on the stream multiprocessors concurrently. The image data is transmitted to page-locked memory after the image is created. Finally, the image data is showed by the display screen. All the operations are scheduled by the threads of CPU.

A NVIDIA GTX 295 GPU is used in this system. In the CUDA model, GPU is treated as a dedicated coprocessor of the host CPU. CUDA allows the same code to be simultaneously run on different cores of GPU as threads.[10,11] All threads are organized into thread blocks, which are executed concurrently on one single-instruction multiple-data (SIMD) stream multiprocessor. A GPU has two types of memory: global memory and stream multiprocessor on-chip memory. Each stream multiprocessor has four different types of on-chip memory: constant cache, texture cache, registers, and shared memory.[10,11] Different types of memory have different latencies. The global memory read function would consume a lot of clock cycles of the GPU, usually 400 to 600 cycles. Thus, to gain optimal performance when utilizing CUDA, the user must determine the thread blocks, the shared memory, registers, and the global memory usage.

As discussed earlier, an interpolation processing only relates to the acquired data and coefficients in the cutoff window, but a lot of interpolation processing use the same acquired data and a lot of lines use the same coefficients. If all the data are stored in global memory, it would consume a lot of read memory time. Thus, it is better to share these data and coefficients by reading them into every block's shared memory before interpolation takes place. The data read speed from the shared memory is almost the same as the speed from the registers, which is the least communication latency in threads.

Furthermore, if a lot of interpolation read the same memory, it would result in block conflicts, which slow down the processing speed. Figure 4 shows the bank conflicts in time-domain interpolation. Figure 4(a) shows an interpolation method that do not use the shared memory. The threads in a block read data from the global memory directly. A data would be read by several threads at the same time, in which bank conflict happens. Another interpolation method which uses the shared memory is showed in Fig. 4(b). A bulk of shared memory is created in a block of GPU, the data is read into the shared memory once, and the threads read the data from the shared memory instead of from the global memory. Thus, no bank conflicts happen.

An experiment is performed to compare the computing cycles between the two methods. Figure 5 shows the results. The cycles increase as N increases, but the cycles of the method without shared memory increase much faster than that of the method with shared memory. The method without shared memory consumes more than four

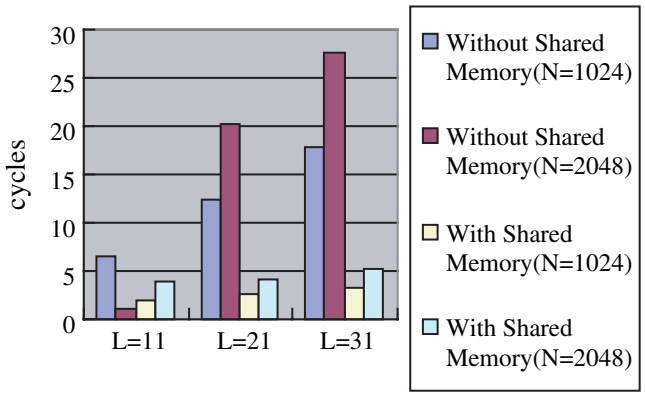

Fig. 5. Comparison between computations with and without shared memory. N is the pixel number. The y-axis displays the operation cycles ($\times 10^5$).

times cycles than the method with shared memory when the $N = 2,048$ and the cutoff length $L = 31$.

A time-domain interpolation algorithm by using shared memory is developed on GPU. Figure 6 shows the algorithm. It follows three steps:

(i) Shared memory is created in every block by using the application programming interface (API) _shared_ of the CUDA model.
(ii) Every block read the acquired data and the coefficients once, reducing the read latency and block conflict.
(iii) Interpolation is processed by multiply–add functions. The resampled data is obtained.

Time-domain interpolation algorithm on GPU
dx=threadIdx.x;
dy=threadIdx.y;
__shared__ float *Ds* [][];
__shared__ float *Cs* [][];
int *begin*;
load the corresponding elements to *Ds* and *Cs* and *begin*;
sum=0;
for(*j*=0; *j*<cut-off length;*j* ++)
sum=*sum*+*Ds*[dx][*begin*]**Cs*[*dy*][*j*];
output *sum*

Fig. 6. The time-domain interpolation algorithm on GPU.

An interpolation processing can be operated by a thread and a lot of threads can be operated concurrently, which is limited by the number of stream multiprocessors in GPU. Thus, GPU can greatly accelerate the computing time.

4. Results and Discussion

To compare the processing capability of the GPU and CPU, the computing time of the steps is measured by CUDA event function. A CUDA event is started and stopped at the beginning and ending, respectively. The computing time of a program is obtained by recording the time between the two events, whose resolution can be reached in microseconds (μs).

An experiment is performed to obtain the processing capability on GPU. Figure 7 shows the results. The computing time can be seen as linear increase as the A-lines increase. The capability of the time-domain interpolation can be reached at 250,000 A-lines, 200,000 A-lines, and 160,000 A-lines

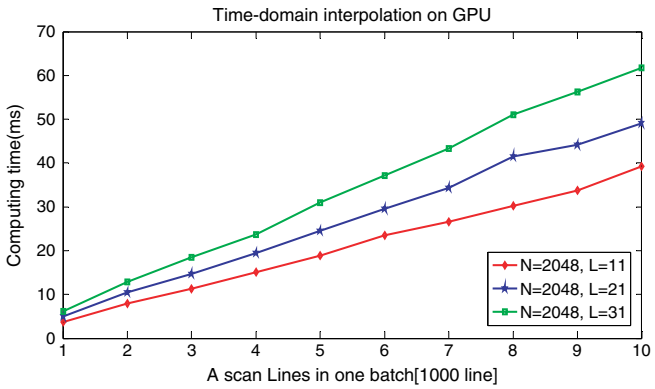

Fig. 7. Time-domain interpolation processing capability on GPU.

Table 1. Computing time of different methods.[a]

	GPU time (ms)	CPU time (ms)	R
Copy data to GPU	0.481	—	—
Background subtraction	2.157	28.60	12.26
Interpolation[b]	1.986	63.21	30.83
FFT	0.844	32.27	37.23
Image creation	0.665	32.12	47.30
Copy data to host	0.090	—	—
Total time	6.223	156.20	24.10

Note: [a]The personal computer platform: CPU, Intel Q9600 2.66 GHz; memory 2G DDR2 800; GPU, NVIDIA GTX 295. [b]The cutoff length is $L = 21$.

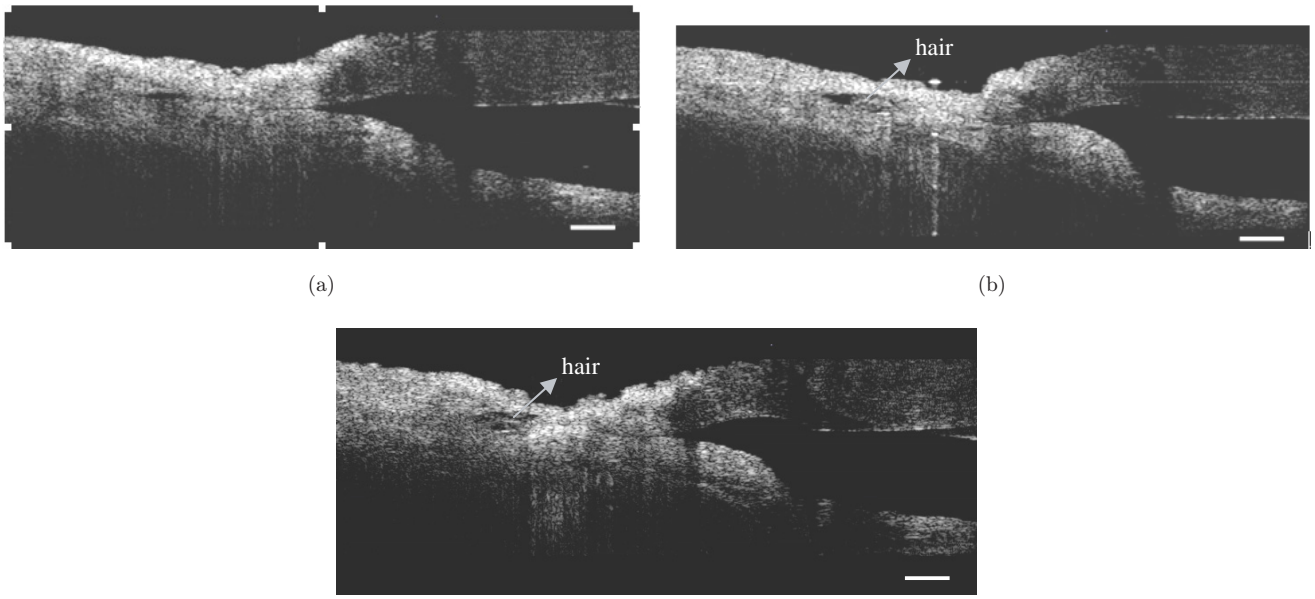

Fig. 8. Different en-face position images of a pig eye. The scale bar is 300 μm.

in a second when the cutoff length is $L = 11$, $L = 21$, and $L = 31$, respectively.

One frame data containing 400 A-lines $\times$ 2,048 pixels $\times$12 bits is acquired and processed. A speed-up ratio (R) is defined as $R = t_c/t_g - 1$, where t_c and t_g are the computing time on CPU and on GPU, respectively. Table 1 shows the results. The computing time of interpolation on GPU ($L = 21$) is much faster than that on CPU, whose R is 30.83. The total signal processing time is 6.223 ms, whose R is 24.10. This speed is much faster than the acquisition speed of the linescan CCD.

To validate the real-time processing capability, a pig eye is scanned. The frame data contain 800 A-lines and the frame rate is 20 frames per second (f/s). The processing can be finished in 10.97 ms when the cutoff window is $L = 21$. The frame rate can be reached in 40 f/s if the scanline is 400, which is limited by the acquisition speed of the linescan CCD. Figure 8 shows the results. A hair was inserted in the pig eye and the viscoelastic was injected to enhance the structure contrast. Different en-face positions are scanned. Different modalities of the eye can be seen in real time. Thus, the real-time processing capability of the SD-OCT system is validated, which is useful in many applications.

In conclusion, we have described a new way to accelerate the computing speed of time-domain interpolation in SD-OCT. More than 250,000, 200,000 and 160,000 A-line can be processed on GPU in one second when the cutoff length is $L = 11$, $L = 21$, and $L = 31$, respectively. The computing time of time-domain interpolation on GPU is 30-fold times less than that on CPU. A frame data containing 400 A-lines $\times$ 2,048 pixels $\times$ 12 bits is acquired and processed. The total signal processing time is 6.223 ms, which has been improved by more than 24 times than that on CPU. Thus, real-time signal processing of SD-OCT is realized by using GPU. Furthermore, CUDA model supports multi-GPU. A higher computing capability can be obtained by using several GPUs.

Acknowledgments

This work is supported by National High Technology R&D project of China (2008AA02Z422) and The Instrument Developing Project of The Chinese Academy of Sciences, Institute of Optics and Electronic, Chinese Academy of Sciences.

References

1. Y. Zhang, X. Li, L. Wei, K. Wang, Z. Ding, G. Shi, *Opt. Lett.* **34**, 1849 (2009).
2. J. G. Fujimoto, *Handbook of Optical Coherence Tomography*, B. E. Bouma, G. J. Tearney, Eds., Chap. 1 (Marcel Dekker, Inc., 2002).

3. R. Leitgeb, C. Hitzenberger, A. Fercher, *Opt. Exp.* **11**, 889−894 (2003).
4. C. Dorrer, N. Belabas, J.-P. Likforman, M. Joffre, *J. Opt. Soc. Am. B* **17**, 1795−1802 (2000).
5. A. W. Schaefer, J. Joshua Reynolds, D. L. Marks, S. A. Boppart, *IEEE Trans. Biomed. Eng.* **51**, 186−190 (2004).
6. S. Yan, D. Piao, Y. Chen, Q. Zhu, *J. Biomed. Opt.* **9**, 454−463 (2004).
7. J. Su, J. Zhang, L. Yu, H. G. Colt, M. Brenner, Z. Chen, *J. Biomed. Opt.* **13**, 030506 (2008).
8. T. E. Ustun, N. V. Iftimia, R. D. Ferguson, D. X. Hammer, *Rev. Sci. Instrum.* **79**, 114301 (2008).
9. G. Liu, J. Zhang, L. Yu, T. Xie, Z. Chen, *Appl. Opt.* **48**, 6365−6370 (2009).
10. NVIDIA CUDA computer unified device architecture: Programming guide, Version 2.0beta2, Jun. 2008.
11. K. Zhang, J. U. Kang, *Opt. Exp.* **18**, 11,772−11,784 (2010).