Performance and Scalability of Fourier Domain Optical Coherence Tomography Acceleration Using Graphics Processing Units

Jian Li,* Pavel Bloch, Jing Xu, Marinko V. Sarunic, and Lesley Shannon

School of Engineering Sciences, Simon Fraser University, V5A 1S6 Burnaby BC, Canada

**Corresponding author: jla193@sfu.ca*

Fourier Domain Optical Coherence Tomography (FD-OCT), provides faster line rates, better resolution, and higher sensitivity for non-invasive, in vivo biomedical imaging compared to traditional Time Domain OCT (TD-OCT). However, because the signal processing for FD-OCT is computationally intensive, real-time FD-OCT applications demand powerful computing platforms to deliver acceptable performance. Graphics Processing Units (GPUs) have been used as co-processors to accelerate FD-OCT by leveraging their relatively simple programming model to exploit thread-level parallelism. Unfortunately, GPUs do not "share" memory with their host processors, requiring additional data transfers between the GPU and CPU. In this paper, we implement a complete FD-OCT accelerator on a consumer grade GPU/CPU platform. Our data acquisition system uses spectrometer based detection and a dual arm interferometer topology with numerical dispersion compensation for retinal imaging. We demonstrate that the maximum line rate is dictated by the memory transfer time and not the processing time due to the GPU platform's memory model. Finally, we discuss how the performance trends of GPU-based accelerators compare to the expected future requirements of FD-OCT data rates.

1. Introduction

Fourier Domain Optical Coherence Tomography (FD-OCT) is a well established optical interferometric imaging modality that has been rapidly gaining popularity for providing high-resolution, cross-sectional imaging in biological tissues and materials. FD-OCT is commonly used in ophthalmic imaging by clinicians and researchers to gain accurate insight into the eyes of patients and research specimen. During the FD-OCT data acquisition, a line scanning camera of the FD-OCT system is used to collect dataset in the form of multiple scanning lines, called amplitude scans (A-Scans). A typical volumetric dataset acquired with currently available commercial systems takes several seconds, while acquisition of more densely sampled volumes for high resolution research applications takes even longer.

The resulting motion artefacts, which are inevitable for live subject imaging, can significantly affect the volumetric image quality and diagnostic utility. In order to achieve minimal motion artefacts, higher acquisition speed is essential. The current fastest line scan camera available for spectrometer based FD-OCT can achieve a line rate of 200kHz with resolution of 4096 pixels [1], and the future trend for acquisition speed is even faster [2]. However, due to the computationally-intensive nature of the FD-OCT algorithm, real-time FD-OCT data processing and displaying is now facing a challenge to keep up with the increasing data acquisition speed of newer cameras. The advances in wavelength swept source OCT (SS-OCT) and parallel FD-OCT systems [3] place even higher demands on computation resources for real time data processing and display of medical images.

Pure software solutions for FD-OCT processing, even on the most up-to-date Intel i7 processors, deliver processing and display rates in the tens of kilohertz. The massively parallel processing cores in a Graphic Processing Unit (GPU) can be utilized to accelerate FD-OCT by off-loading the computationally-intensive processing tasks from the CPU. Previous work [4] reported the use of a NVIDIA GTX285 as an accelerator for FD-OCT using a k-linear spectrometer, a configuration that is not commonly used in commercial FD-OCT systems. A follow up study presented GPU acceleration for a standard configuration spectrometer using a high end NVIDIA FX5800 GPU [5]. The authors reported processing rates for a common path interferometer topology, excluding the additional processing steps required to compensate for the dispersion present in the more common dual-arm interferometer configurations used in ophthalmic imaging [2]. GPUs are also used as accelerators in Monte Carlo simulation [6, 7], a recent study [8] demonstrated a speedup of approximately 600 times over Intel i7 CPU for Monte Carlo simulation on the new Fermi GPU with a novel scheme to optimize memory utilization.

In this paper, we investigate the limitations of GPU-based accelerators for real time FD-OCT processing and project how current trends in GPU architecture will scale for increasing line rates. Specifically, the contributions of this paper are:

- a GPU accelerated software implementation of a complete FD-OCT system; we demonstrate that the GPUs are able to process at a maximum line rate of 280 kHz with dispersion compensation, whereas the maximum throughput for the overall system is limited to 110 kHz due to the overhead incurred by data transfers.
- the GPU accelerated FD-OCT system incorporates the dispersion compensation pro-

cessing [2] required for dual arm interferometers and retinal imaging,

- an analysis of the FD-OCT algorithm's performance profile on a GPU and the implications of current trends in GPU architecture for future line rates.

The paper is structured as follows: Section 2 summarizes the related previous work; Section 3 describes the data acquisition system and processing algorithm; results are given in Section 4, and finally we conclude the paper and discuss future work in Section 5.

2. Background

In FD-OCT processing, each A-Scan is independent of one another. This characteristic of FD-OCT suggests speedups could be achieved by exploiting parallelism in the following aspects,

- FD-OCT A-Scans will benefit from Single Instruction Multiple Data (SIMD) processing from GPU as it is in the form of vectors,
- multiple A-Scan can be processed in parallel because individual lines are independent, and
- in the case where the vector operations involves constants, each *element* within an A-Scan vector could be processed in a pipelined fashion.

Watanabe et. al demonstrated the use of a NVIDIA GTX285 GPU to accelerate FD-OCT systems with a line scan CCD camera at 27.9 kHz, while achieving a real-time display frame rate of 27.9 frames/sec (2048 FFT size, 1000 A-Scans) [4, 9]. The calculation of Fourier transforms from wavelength space to wave number space is avoided, because a k-linear spectrometer was used. A follow-up study [10] from the same group used a linear wavelength (λ) spectrometer with a zero-filling interpolation technique. Zhang et. al reported a standard spectrometer configuration with a common-path interferometer topology [5], therefore excluding the dispersion compensation step required in the more common dual-arm interferometer setup. This work implemented both the λ-to-k resampling and 3D real-time volumetric rendering on a FX5800 GPU, however, only the GPU processing time was used to evaluate the system performance. Columns 2 and 3 of Table 1 list the specifications of the GPUs used in the previous works.

In this paper, we investigate the performance of a GPU-accelerated FD-OCT system under imaging conditions using a dual-arm interferometer topology. It should be noted that while GPU accelerated platforms deliver significant speed up by parallelizing the FD-OCT algorithm, they also require additional steps to copy data back and forth between the host (CPU) and device (GPU) memory; when calculating the maximum real-time throughput rate of GPU accelerated platforms, the time required for these additional memory copies needs to be included.

The system presented is based on a commercial spectrometer customized by Bioptigen Inc. As will be discussed in Section 3.B, in addition to all the processing functions in [5], our system also included a numerical dispersion compensation function, which is highly computationally intensive as it uses two Fourier transforms and vector multiplication in complex numbers. For this reason, a higher computational requirement has to be met in order to achieve the goal of real-time FD-OCT processing and display. In addition to implementing the GPU accelerator, we also quantified the actual runtime of each of the processing steps of the FD-OCT algorithm so as to gain better insight into the performance limiting aspects of the algorithm on the GPU accelerating platform; this information can then be used to guide future research into accelerating the processing of FD-OCT for real-time applications. As demonstrated by our analysis, the data transfer rate between CPU and GPU is currently the limiting factor to accelerating the FD-OCT algorithm.

3. An Overview of a FD-OCT System

A typical FD-OCT system is composed of three stages — data acquisition, processing and display, as noted in Figure 1.

3.A. Data Acquisition

During the data acquisition, a line scanning camera acquires a dataset from the spectrometer, which, depending on the configuration of the acquisition device, could be either collected to form a frame using a frame-grabber for batch transfer or directly transferred out in a serial fashion (via interfaces such as CamerLink). The demands for faster processing stems from the increasing acquisition speed, which is in turn made possible by faster scanning rate and higher resolution. While the scanning rate on a single channel FD-OCT acquisition on ophthalmic imaging will be bounded for safety reasons, multi-channel scanning implementations could easily scale the equivalent speed up to 6 times [3].

3.B. FD-OCT Processing

During the processing stage, the data set is processed through the following functions. The details of the algorithm are described in [2] and [5].

3.B.1. DC Removal

This function subtracts the DC level. While the average DC levels (shown as *Avg. DC*) will need to be acquired (involving add/accumulation/division) prior to the removal, the DC level remains relatively constant for a specific acquisition, thus it is not necessary to perform DC acquisition in real time. The DC removal functions is composed of vector subtraction.

3.B.2. λ-to-k Re-sampling

Because the majority of line scanning cameras deliver data sets that are sampled in wavelength (λ) space, whereas the FD-OCT algorithm requires data to be presented in evenly sampled wave-number (k) space, a linear-k space data set needs to be constructed from the one based on linear-λ space [11]. Linear interpolation is used to *re-sample* the required data set [5]. As the sampling frequency and its range (shown as *Freq.& Range* in Figure 1) for a specific image acquisition is fixed, the coefficients for linear interpolation can be obtained prior to the real time FD-OCT processing. The λ-to-k re-sampling function contains vector add, multiplication, as well as re-ordering.

3.B.3. Dispersion Compensation

This function is an extra step to the FD-OCT processing and is required for high quality imaging in applications such as retinal imaging [2]. The components of this function contains a Hilbert Transform [2, 12], which is used to construct a complex number representation of the spectral sampled data. The compensation is achieved by tuning the phase coefficients up to the third term [2, 11] (shown as *Phase Coe.* in Figure 1). The Hilbert Transform is implemented by a forward Fast Fourier Transform (FFT), vector multiplication in complex numbers and an inverse FFT.

3.B.4. FFT

A cross sectional image is produced by resolving the information on delay and magnitude of the optical reflection. This information can be extracted by performing an FFT of the spectral interference signal the length of the FFT is 2048.

4. Experimental Setup

Figure 3 shows the block diagram of the FD-OCT system used in this work. A Dalsa Spyder II camera, with 2048 pixels and a maximum line rate of 36 kHz, is used as the detector for data acquisition. The camera communicates with the computer via gigabit Ethernet and transmits entire frames (as opposed to individual line scans). Ideally, acquisition of the next frame is performed at the same time the current frame is being processed.

The GPU accelerator used for our implementation was a dual-GPU GeForce GTX295 from NVIDIA (shown in column 4 of Table 1), with 480 CUDA processor cores (240 cores per GPU) and 17892 MB of DDR3 memory (896 MB per GPU), and a core clock frequency of 576 MHz. For the current implementation, we are using one GPU (240 CUDA cores and 896 MB of GPU memory) to make a fair comparison to previous works [4, 5]. The GTX295 GPU board communicates with the CPU via the 16 lanes PCI-Express v2.0 bus as in the comparative works.

We used the Compute Unified Device Architecture (CUDA) language [13], an extension of the C language for programming NVIDIA GPUs (version 2.3), to develop and debug the FD-OCT algorithm on the GTX295 GPU. Figure 2a shows the algorithm flowchart, as well as the data flow between the host (CPU and main memory) and the device (GPU and device memory). Whenever one frame of data is available in the main memory, the whole frame is transferred over to the GPU and processed as a batch. The CUFFT library from NVIDIA [14] is used to process the FFT in two of the processing steps, specifically the Dispersion Compensation and FFT2048 steps. On the completion of the processing on the GPU, the results are copied back to the CPU for rendering and display[1]. We measured the total FD-OCT processing time including the memory transfer time between host and device using *cudaprof* [15], the program profiler provided by NVIDIA. Figure 2b shows a sample retina image with a resolution of 1024x512 processed using our system.

5. Results and Discussions

In this section we present an analysis of the FD-OCT algorithm on GPUs in terms of its execution profile and throughput bottlenecks, we then look into the scalability of both platforms for future line rates.

5.A. Algorithm Execution Profile Analysis

Figure 4 illustrates the percentage of the FD-OCT algorithm's runtime as attributed to its various component functions (similar results was seen in [9] which didn't include dispersion compensation). Although the specific results presented here are for processing multiple A-scans in a batch, the percentages of time allocated to the various functions remain relatively constant as long as the size of the data set is large enough to amortize the cost of the data transfer[2]. Figure 4 shows that the processing steps for the FD-OCT algorithm (DC-Removal, Resample, Dispersion Compensation, FFT2048 and Logarithm) require approximately 40% of the total runtime time, while the memory (data) transfers (host-to-device and device-to-host) require approximately 60%. The maximum processing line-rate, excluding the dispersion compensation and data transfer, was ∼680 kHz. Adding in the time required to perform the computationally heavy dispersion compensation slowed the performance down to ∼280 kHz. However, to properly evaluate the real-time performance of a GPU accelerated implementation, the data transfer time must also be included as it is an unavoidable overhead for existing GPU architectures. When the time for data transfers between the CPU and GPU are

[1] Ideally, the post-processed data should be directly copied into the frame buffer on the GPU for display without additional copies between host and device.

[2] Due to the overhead incurred from initiating data transfers between device and host memory, data copies need to be "batched" (i.e. multiple individual copies grouped together into a single multi-word copy) to amortize this cost [13]. In Figure 4, a batch size of 8192 was used.

included, the resulting maximum camera line rate for the complete system implementation is approximately $\sim$110 kHz.

Figure 5 shows the performance of the GPU accelerated implementation in terms of processed A-scan lines per second versus varied batch sizes for the three different scenarios indicated in the legend: 1) *Memcpy Excl*, the line rate for pure processing without the memory copies; 2) *Memcpy Incl*, the line rate including both memory copies (via a 16-lane PCI Express Bus); and 3) *Intg. GPU*, the extrapolated line rate for processing plus memory transfers if the same GPU architecture had an integrated configuration.

Although NVIDIA recommends maximizing the size of the data transfers to minimize the overhead, Figure 5 illustrates that there is little difference in the system's actual maximum line rate (*Memcpy Incl*) for varied batch sizes due to the fact that the majority of the "processing" time ($\sim$60%) is spent in memory copies. The trend line excluding the memory transfers (*Memcpy Excl*) demonstrates that when only pure GPU processing time is considered, there is greater variation in line rates over varied batch sizes, the trend line plateaus after reaching maximum at 8192 A-Scans. Finally, while the GPU used in these experiments is only available as a discrete unit due to the processing power requirements, we projected what the maximum line rates would be if it were available as an integrated device[3]. We measured the time needed for data transfers between a CPU and an integrated NVIDIA GeForce 9400M GPU, which shares the same discrete memory model where the CPU data must be transferred to the "device memory" of the GPU (even when the host and device memory are on the same physical device). We used this information to extrapolate what the effective line rate would be for our GPU in this configuration (*Intg. GPU*). Surprisingly, it is the memory copy itself, and not the data transfer via the PCI express bus, that accounts for the majority of the performance loss of relative to the pure GPU processing time (*Memcpy Excl*). In fact, the integrated GPU scenario only improves the effective line rate of our actual system by 22%.

Therefore, to significantly increase the maximum line rate, the underlying memory architecture of the GPU needs to be changed to limit the number of memory transfers between the CPU and GPU. GPUs on high end work stations with higher PCI Express bus frequency will provide better processing rates for FD-OCT, but it is still necessary for data to be transferred from the CPU for processing. However, changes to the GPU memory architecture would allow processed data to be directly rendered to the display and significantly impact the maximum line rate of the system (nominally a 1.3x increase). We are currently investigating the latest Fermi architecture GTX480 GPU (shown in column 5 of Table 1)

[3]Integrated GPUs are packaged on the same chip as the system memory controller and system I/O controller to provide a compact, low cost, low power-consumption solution. Due to these design constraints, they provide less processing power and fewer processing cores to meet the requirements.

from NVIDIA that supports duplex data transfers between the host and device with the goal of reducing the impact of the memory copy by writing post-processed lines back to the CPU while new data for processing is read onto the GPU. Furthermore, we are pursuing the use of multiple host threads to exploit the additional spatial parallelism available on multi-GPU devices so that one device renders and displays the results while the other device processes new scans. In general, the key to improving the throughput rates for real time processing of increased line rates will be reliant on leveraging increased spatial parallelism in the algorithm. As line rates increase, assuming no dramatic changes in GPU architectures, this may require the use of hardware acceleration to achieve this increased spatial parallelism.

In summary, whereas the GPU processing has a line rate of $\sim$280 kHz, the system's effective line rate is $\sim$110 kHz. To compare our implementation to previous work, we exclude the data transfer time as well as the dispersion compensation function, resulting in a maximum line rate of $\sim$680 kHz. By comparison, the implementation of [5] has a reported maximum line rate of $\sim$320 kHz on the FX5800 GPU, demonstrating that our implementation achieves faster line rate on a GPU with the same number of processing cores but less memory and a slower clock rate.

5.B. Scalability of Using GPUs as FD-OCT Accelerators

While technology sizes continue to decrease dramatically, following the trends predicted by Moore's law [16], clock speeds have leveled off due to power consumption and heat dissipation requirements. Thus, improved computing power is provided in terms of spatial processing, or additional processing units. This additional processing power is extremely beneficial to data processing applications with independent data sets that can be processed in parallel, as in the case of FD-OCT, where each A-Scan can be processed independently. Ideally, as the number of processing cores increases, larger batch sizes can be processed in parallel to increase the throughput.

Figure 6 graphs the maximum processing line rate (i.e. excluding the time for memory transfers) of the GPU used in the previous experiment, the GTX295, as well as the newer GTX480. Recalling Table 1, the GTX480 has twice as many processing cores as the GTX295. As can be seen in Figure 6, the GTX480, has a maximum line rate approximately 2.2x that of the GTX295, demonstrating the algorithm's ability to leverage the additional processing cores on the GTX480. However, Figure 6 demonstrates that the maximum line rate trend lines for both devices are consistent with each other and flatten out at 2048 A-scans.

Future high speed data acquisition rates for FD-OCT could reach up to 1 MHz for cardiovascular imaging, while multi-channel FD-OCT system could easily have an even higher demand [17]. Considering the performance of our current GPU accelerated FD-OCT system of about 280 kHz, at least a three times speed up is required to meet these future demands.

Although Figure 6 demonstrates that the number of processing cores on the GPU may soon meet this need, as the memory transfers consume most of the system accelerator's time (∼60%), it will not be possible for GPUs with the current memory architecture to scale to this line rate.

6. Conclusions

In this paper, we discussed a GPU accelerated real-time FD-OCT system implementation with numerical dispersion compensation that achieved a maximum overall line rate of ∼110 kHz including the time required for data transfer, with the maximum processing-only line rate being ∼280 kHz. We analyzed the algorithm's performance profile to determine its breakdown in terms of the GPU's processing time and demonstrated that the performance bottleneck of using the GPU as an accelerator is the data transfer time between host and device memory-independent of whether or not the GPU is on a PCI Express card or integrated on the motherboard. As the clock rates of future devices will not increase significantly, reducing the data transfer time between the CPU and the GPU will have the greatest impact on throughput. In fact, enabling these GPUs to directly render their processed data to the display (as opposed to requiring additional memory transfers) will result in the greatest increase in line rate. This better leverages the GPU architecture's best asset - spatial parallelism.

Assuming future GPU architectures do not require memory copies, researchers will need to determine if the latency of GPU accelerated systems is appropriate for real time applications, including micro-surgeries, particularly if volumetric rendering of the structures is desired. As the GPU based system achieves high throughput by processing multiple frames at once, it may not meet the desired latency specifications to enable appropriate response times. Therefore, platforms such as Field Programmable Gate Arrays, which allow a finer grain of parallelism and pipelining such that individual A-scans can be processed on arrival, should be investigated to determine if they better meet the long term scalability requirements of future data acquisition rates and real time response times.

Acknowledgments

We would like to thank the Canadian National Science and Engineering Research Council (NSERC), the Canadian Institute of Health Research (CIHR) and the Michael Smith Foundation for Health Research (MSFHR) for funding this project. We would also like to thank the Canadian Microelectronics Corporation (CMC) for their equipment donations.

References

1. T. Schmoll, C. Kolbitsch, and R. A. Leitgeb, "Ultra-high-speed volumetric tomography of human retinal blood flow," Opt. Express **17**, 4166–4176 (2009).
2. M. Wojtkowski, V. Srinivasan, T. Ko, J. Fujimoto, A. Kowalczyk, and J. Duker, "Ultrahigh-resolution, high-speed, Fourier domain optical coherence tomography and methods for dispersion compensation," Opt. Express **12**, 2404–2422 (2004).
3. M. K. K. Leung, A. Mariampillai, B. A. Standish, K. K. C. Lee, N. R. Munce, I. A. Vitkin, and V. X. D. Yang, "High-power wavelength-swept laser in littman telescope-less polygon filter and dual-amplifier configuration for multichannel optical coherence tomography," Opt. Lett. **34**, 2814–2816 (2009).
4. Y. Watanabe and T. Itagaki, "Real-time display on Fourier domain optical coherence tomography system using a graphics processing unit," Journal of Biomedical Optics **14**, 060506 (2009).
5. K. Zhang and J. U. Kang, "Real-time 4D signal processing and visualization using graphics processing unit on a regular nonlinear-k Fourier-domain OCT system," Opt. Express **18**, 11772–11784 (2010).
6. Q. Fang and D. A. Boas, "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units," Opt. Express **17**, 20178–20190 (2009).
7. N. Ren, J. Liang, X. Qu, J. Li, B. Lu, and J. Tian, "GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues," Opt. Express **18**, 6811–6823 (2010).
8. E. Alerstam, W. C. Y. Lo, T. D. Han, J. Rose, S. Andersson-Engels, and L. Lilge, "Next-generation acceleration and code optimization for light transport in turbid media using GPUs," Biomed. Opt. Express **1**, 658–675 (2010).
9. Y. Watanabe and T. Itagaki, "Real-time display on SD-OCT using a linear-in-wavenumber spectrometer and a graphics processing unit," in "Optical Coherence Tomography and Coherence Domain Optical Methods in Biomedicine XIV," , vol. 7554, J. A. Izatt, J. G. Fujimoto, and V. V. Tuchin, eds. (SPIE, 2010), vol. 7554, p. 75542S.
10. Y. Watanabe, S. Maeno, K. Aoshima, H. Hasegawa, and H. Koseki, "Real-time processing for full-range Fourier-domain optical-coherence tomography with zero-filling interpolation using multiple graphic processing units," Appl. Opt. **49**, 4756–4762 (2010).
11. J. Xu, L. Molday, R. Molday, and M. Sarunic, "In vivo imaging of the mouse model of X-linked juvenile retinoschisis with Fourier domain optical coherence tomography," Investigative ophthalmology & visual science **50**, 2989 (2009).
12. J. Goodman, *Statistical Optics* (Wiley, New York, 2000).
13. "NVIDIA CUDA Programming Guide," (2009), http://developer.nvidia.com/object/cuda_2_3_downloads.html.
14. "CUDA CUFFT Library," (2009), http://developer.nvidia.com/object/cuda_2_3_downloads.html.
15. "CUDA Visual Profiler," (2009), http://developer.nvidia.com/object/cuda_2_3_downloads.html.

16. G. Moore *et al.*, “Cramming more components onto integrated circuits,” Proceedings of the IEEE **86**, 82–85 (1998).

17. W. Wieser, B. R. Biedermann, T. Klein, C. M. Eigenwillig, and R. Huber, “Multi-megahertz OCT: High quality 3d imaging at 20 million a-scans and 4.5 gvoxels per second,” Opt. Express **18**, 14685–14704 (2010).

List of Tables

List of Figures

Table 1. Specifications for the GPU in discussion.

GPU Models	FX5800 [5]	GTX285 [4]	GTX295	GTX480
Processing Cores	240	240	2x240	480
Device Memory (MB)	4096	2048	2x898	1536
Core Frequency (MHz)	610	648	576	700
Shader Frequency (MHz)	1296	1476	1242	1407
Memory Frequency (MHz)	800	1242	999	1848
Price (USD)	$\sim$3000	$\sim$500	$\sim$500	$\sim$500

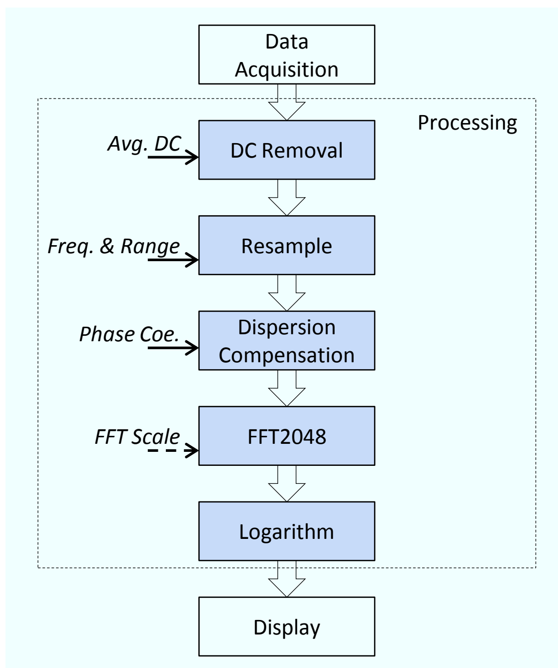
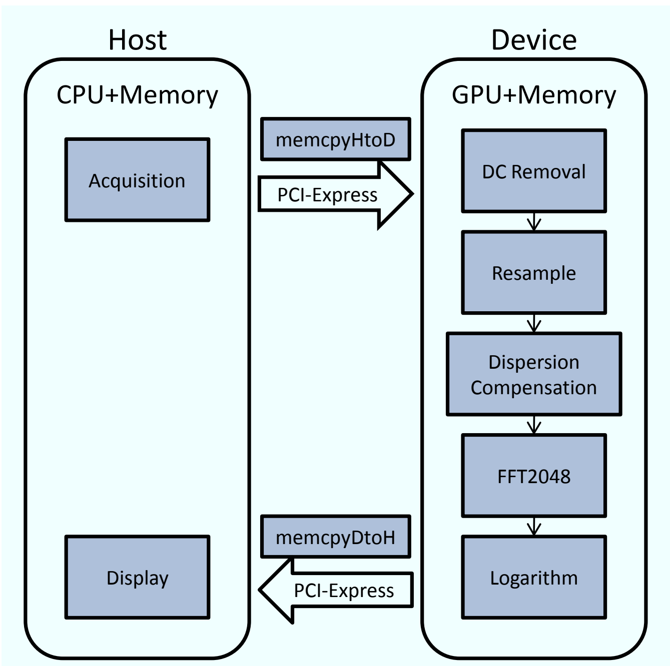


Fig. 1. The FD-OCT processing flow.



(a) GPU flowchart

(b) A sample retina image

Fig. 2. Algorithm flowchart for GPU and a sample retina image.

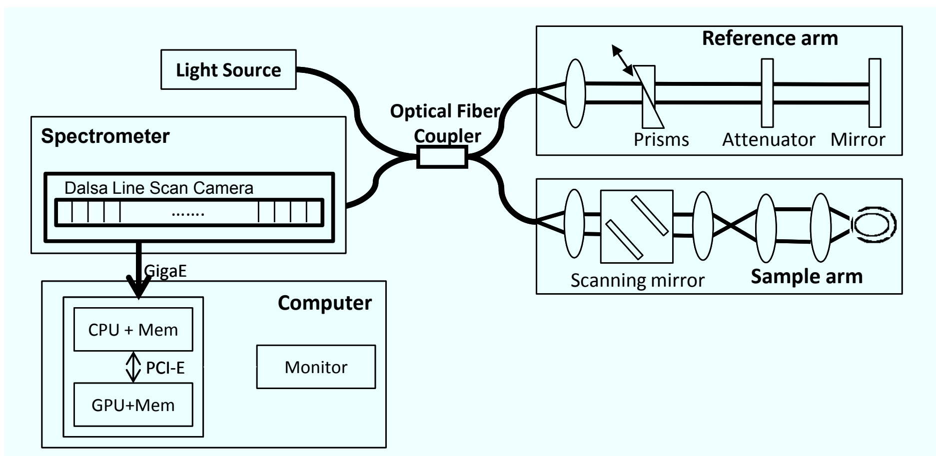


Fig. 3. The complete FD-OCT system using GPU as co-processor.

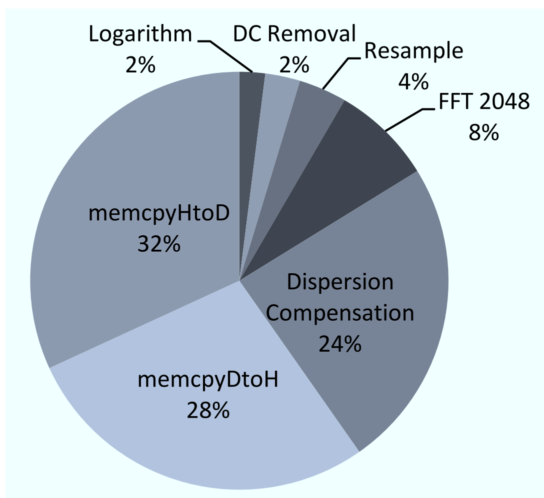


Fig. 4. The percentage of the GPU functions runtime.

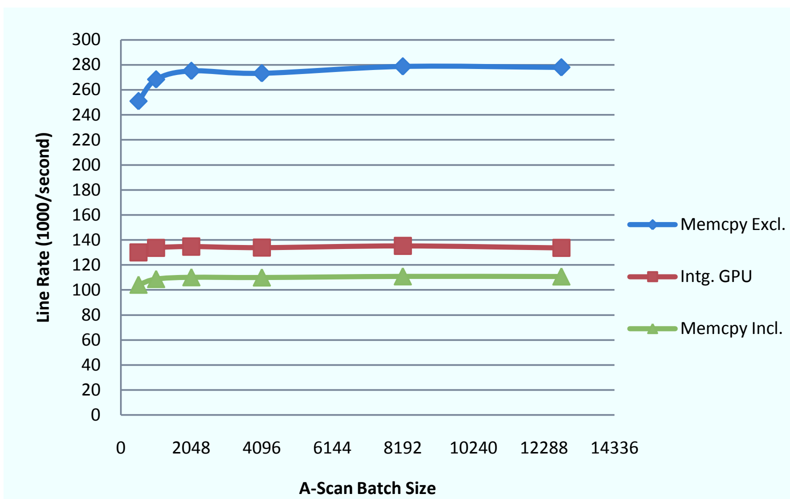


Fig. 5. Performance against A-Scan batch sizes.

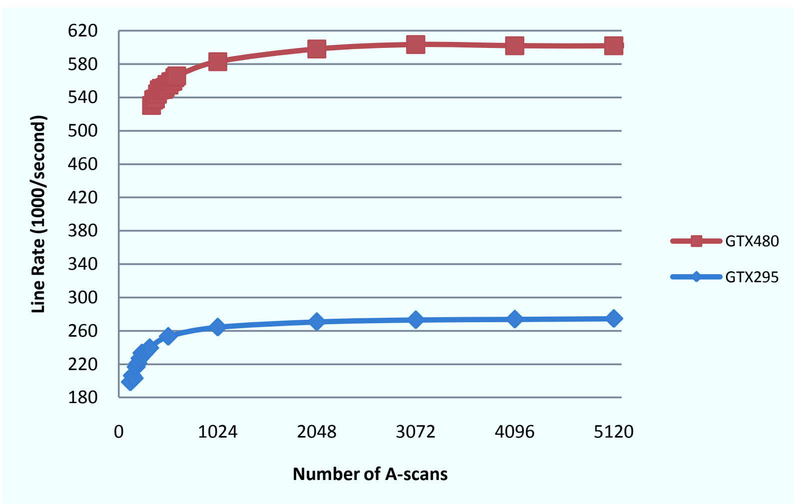


Fig. 6. Processing-only line rate over different A-Scan on two GPUs.