

Using GPUs for Image Processing

Ben Baker, FamilySearch
bakerb@familysearch.org

Abstract

Graphics Processing Units (GPUs) have been traditionally used to accelerate computation of computer graphics in applications such as video gaming and high-end 3D rendering. However, recent research has examined using GPUs “in reverse” [1] for computer vision types of image processing. This paper examines leveraging the parallel processing capabilities of GPUs to lower costs and increase the throughput of the millions of original record images being processed by FamilySearch.

Digitization of original records is a large focus of family history service providers such as FamilySearch. This digitization enables family history researchers to more easily access images of records without requiring physical access to archives or microfilm. After digital images of records have been captured, FamilySearch applies several treatments to the raw images to produce both preservation and distribution quality images.

Examples of these treatments include decoding from and encoding into different image formats, automatic skew correction, automatic document cropping, image sharpening and image scaling to produce thumbnail images. The intent of these treatments is to enhance the presentation of the records in the images to the end user and to reduce file size for storage.

FamilySearch currently processes millions of images annually in this manner through a collection of CPU based servers called the Digital Processing Center (DPC). While the DPC consists of many CPU cores running in parallel across multiple servers, recent GPUs include comparable numbers of less powerful cores in a single card.

If servers are constructed with both CPUs and GPUs and code is written to utilize the multitude of cores on the GPUs in a parallel manner, comparable throughput may be achieved in a smaller form factor with less overall cost and decreased processing time per image. The result is increased scalability as FamilySearch continues to increase the number of images processed to make more records available more quickly to more people.

1. Background on GPU Computing

As the graphical capabilities of personal computers have increased, hardware manufacturers discovered that there were advantages to creating specialized hardware to perform mathematical operations commonly used in graphics rendering. This eventually resulted in the advent of modern video cards with Graphics Processing Units (GPUs) as their core processing units.

In contrast, Central Processing Units (CPUs) are general-purpose processors capable of running many different applications. The architectures of CPUs and GPUs have evolved over time, but have always had fundamental differences. CPUs have been optimized to provide a high degree

of instruction level parallelism to maximize performance. In addition, recent CPUs have provided multiple processing cores further allowing data to be processed in parallel.

On the other hand, GPUs utilize a highly parallel architecture composed of many more but smaller processing elements capable of a high degree of data level parallelism. GPUs are very much designed to be single instruction, multiple data (SIMD) machines. This was originally for the purpose of accessing multiple pixels simultaneously to improve computer graphics performance.

Modern GPU hardware is already fast and getting faster more quickly than CPU hardware as shown by the growth curves in Figure 1 [2]. Despite higher processing capabilities, GPU computation is not well suited for certain types of tasks, however. In order to realize performance gains from GPU computing, algorithms must be written in a manner that takes advantage of and executes on the GPU instead of the CPU.

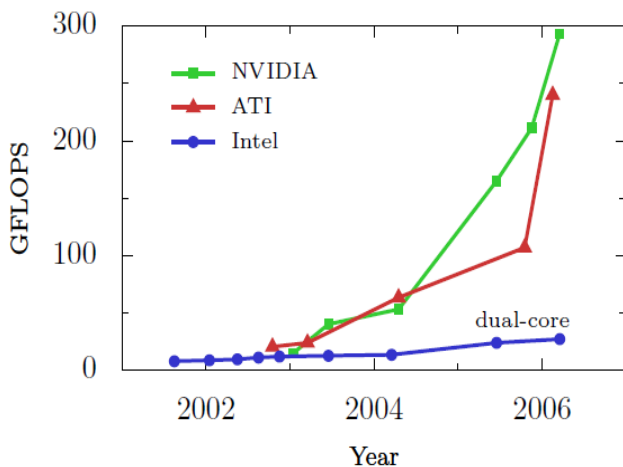


Figure 1 – GPU processing capabilities have grown much more quickly than for CPUs [2].

Despite the original intention of GPUs being used as graphics processors, advances in GPU architectures, programming tools and languages have given rise to the General Purpose GPU field [3]. In the past, specialized knowledge and skills were required to effectively write code designed for GPUs and applications were somewhat limited to the graphics domain. However, programming tools have now been developed to use GPUs as general purpose parallel processors by writing code in high level programming languages in a way familiar to the majority of software developers [4].

As an example, one of the leading GPU hardware manufacturers, NVIDIA, has developed a programming language similar to the C programming language based on their Compute Unified Device Architecture (CUDA) [5]. Language bindings for CUDA have also been developed for many other popular programming languages, making writing code to utilize the computational power of GPUs much easier. As evidence of the growing popularity of parallel programming on GPUs, parallel programming courses using CUDA are being taught at 381 universities throughout the world [6].

Improved programming languages and tools have given rise to the emerging domain of High Performance Computing (HPC) with GPUs separate from the more traditional markets of high-end computer gaming and 3D rendering. New products such as the NVIDIA Tesla [7] are specifically designed solely for computation as evidenced by the complete lack of video output like a traditional video card. These GPU computing processors are being used in large clusters for weather simulations, medical imaging, computational finance and many more computationally intensive industries. NVIDIA also promotes using these cards to create “personal supercomputers”, capable of computation far exceeding normal workstations.

NVIDIA claims that the latest Tesla 20-series GPU computing processors can deliver equivalent performance to the latest quad-core CPUs at 1/20th the power consumption and 1/10th the cost [7] in smaller form factors. These aspects make using GPUs for image processing at FamilySearch an attractive option to improve throughput and reduce ongoing costs.

2. Background on FamilySearch Digital Image Processing

FamilySearch is an organization that obtains a large amount of digital images of genealogical records from microfilm and digital capture of original records. These images are given various processing treatments to enhance the presentation of the records in the images and reduce file size for storage.

This image processing takes place in a collection of several CPU based servers called the Digital Processing Center (DPC). FamilySearch currently processes millions of images annually with projections of at least a two-fold increase year over year for the next two years. Future projections also include more color image processing (the vast majority is currently 8-bit grayscale) and larger images from digital cameras, further increasing processing requirements.

FamilySearch currently accelerates image processing by leveraging two widely used libraries for a large portion of the image processing performed in the DPC. These libraries are Intel's Integrated Performance Primitives (IPP) [8] [9] and OpenCV [10] libraries. In addition to these image processing libraries, several well used image decoder/encoder libraries are used to convert images from one format to another for various purposes.

The majority of the time spent during processing each image occurs during relatively few library calls in the aforementioned libraries, so finding a way to replace these library calls with faster parallel GPU based alternatives would directly translate to improved overall image processing throughput in the DPC.

3. Potential GPU Based Solutions

Since the majority of time spent on image processing computation in the DPC takes place in library function calls to the IPP and OpenCV libraries, finding comparable GPU based libraries for these methods would be very advantageous. This approach relieves developers from the optimization burden and learning curve of programming directly on the GPU in CUDA or similar languages.

In direct response to Intel's IPP library, NVIDIA has produced a NVIDIA Performance Primitives (NPP) library [11] [12]. The latest version (3.2.7) of this library has implemented several hundred functions that correspond to IPP library functions. The intention for this library is to provide a GPU based solution that could be integrated easily with existing projects utilizing IPP, such as FamilySearch's image processing library.

NVIDIA has also recently partnered with Willow Garage, the maintainers of OpenCV, to include a GPU module that provides acceleration of some library functions on GPU hardware in the latest release of OpenCV (2.2) in December 2010. This module is admittedly in early beta stage,

but is suitable for this investigation. Another potential candidate to accelerate OpenCV functions on the GPU is GPUCV [13].

The current architecture of FamilySearch's image processing library is such that it should be possible to augment the current CPU based DPC cluster with GPU based servers that are capable of utilizing code that calls into libraries that execute on these GPUs. If the GPU based cluster's performance is favorable, it may be possible to eventually replace the current CPU based cluster with a GPU based one controlled by far fewer CPU based servers. Assuming sufficient speedup is obtained, comparable throughput would be achieved in a smaller form factor with lower power consumption and total cost of ownership.

4. Performance Testing Methodology

To show the viability of GPU based image processing at FamilySearch, this paper will examine two image processing operations currently performed by the DPC, cropping and sharpening. Performance of the current CPU based library will be compared against a GPU based prototype to illustrate the performance gains as well as limitations of image processing on the GPU.

All performance tests were executed on a system with Dual Quad Core Intel® Xeon® 2.83GHz E5440 CPUs (8 cores total), 16 GB RAM and a Debian Linux operating system. A single Tesla C1060 Compute Processor (240 processing cores total) was connected via a PCI-Express x16 Gen2 slot for all GPU computations.

Three representative images of increasing size were chosen as samples for performance testing purposes. The smallest image of 1726 x 1450 (2.5 megapixels) represents one of the smaller images processed by FamilySearch. A larger 4808 x 3940 (18.9 megapixel) image represents a relatively typical size of image and the largest 8966 x 6132 (55.0 megapixel) image represents the current maximum size of images processed by FamilySearch.

5. Cropping Operation Results

Images are cropped by FamilySearch to provide a uniform border around records within images throughout a collection and to save storage space by discarding pixels that do not provide relevant information pertinent to the record. This is also important when the images are indexed by FamilySearch Internet Indexing [14] so that a template can be applied to images that will more accurately place highlights on fields with data that should be indexed.

The cropping operation consists of three main steps:

1. Compute a threshold value
2. Binarize the image based on the computed threshold
3. Compute a bounding box that encloses all pixels determined as part of the document

The operation to binarize the image based on a threshold value is well suited to parallelization on the GPU. The NPP library provides the `nppiThreshold` and `nppiCompare` methods that are capable of performing this operation. Computing a threshold value requires creating a histogram of the image, which can also be optimized by using NPP functions.

Results comparing the time to compute a threshold and binarize the image are shown in Figure 2. Performing the threshold computation and binarization portions of the crop operation on each image using the current image processing library in use by FamilySearch closely follows a linear trajectory of about 12ms per megapixel in the image.

Image Name	Megapixels	CPU	GPU	Speedup
MARY0004D_Estate-Records-6456_06464-bl.tif	2.5	28.5	25.0	1.14
SAM0203_00013.tif	18.9	213.0	30.0	7.10
ITALRO02D_1-7-37_lsi.tif	55.0	640.5	40.0	16.01

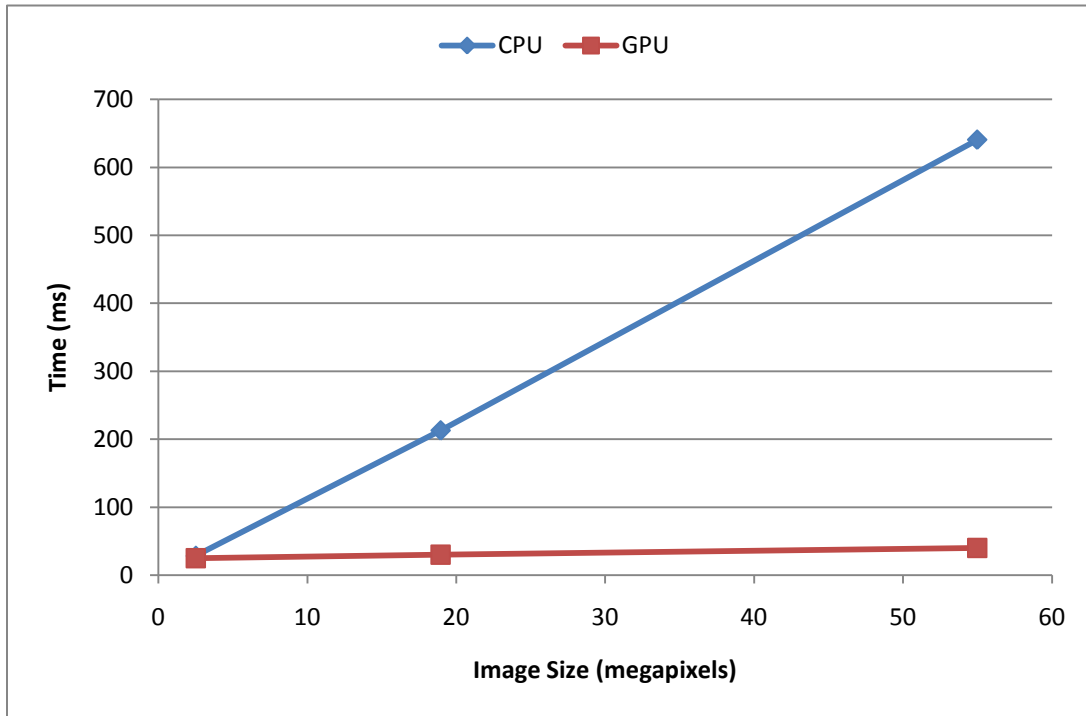


Figure 2 – Comparison of time to compute threshold and binarize images

Using the NPP library methods to threshold the same images easily shows the effects of parallelization. For the smallest image, the time to perform the binary thresholding on the GPU was only 14% faster than the CPU implementation. As image size increases, however, the time to compute the threshold and binarize only increases slightly. The largest test image exhibited a 16x speed increase when performed on the GPU over the CPU implementation.

Not all portions of the crop operation were effectively parallelized on the GPU, however. Although NPP histogram functions were used to assist with computing a threshold value, the actual threshold calculation was still performed on the CPU. This was not a significant amount of time, however, since this time was represented in results found in Figure 2. The code to compute the bounding box also remained on the CPU and did not benefit from parallelization. If the time for the entire crop operation is accounted for, the overall speedup was nonexistent or much smaller depending on image size. See Figure 3 for a comparison of the overall time to crop images.

Image Name	Megapixels	CPU	GPU	Speedup
MARY0004D_Estate-Records-6456_06464-bl.tif	2.5	73.3	75.3	0.97
SAM0203_00013.tif	18.9	427.0	406.0	1.05
ITALRO02D_1-7-37_lsi.tif	55.0	1285.7	1131.0	1.14

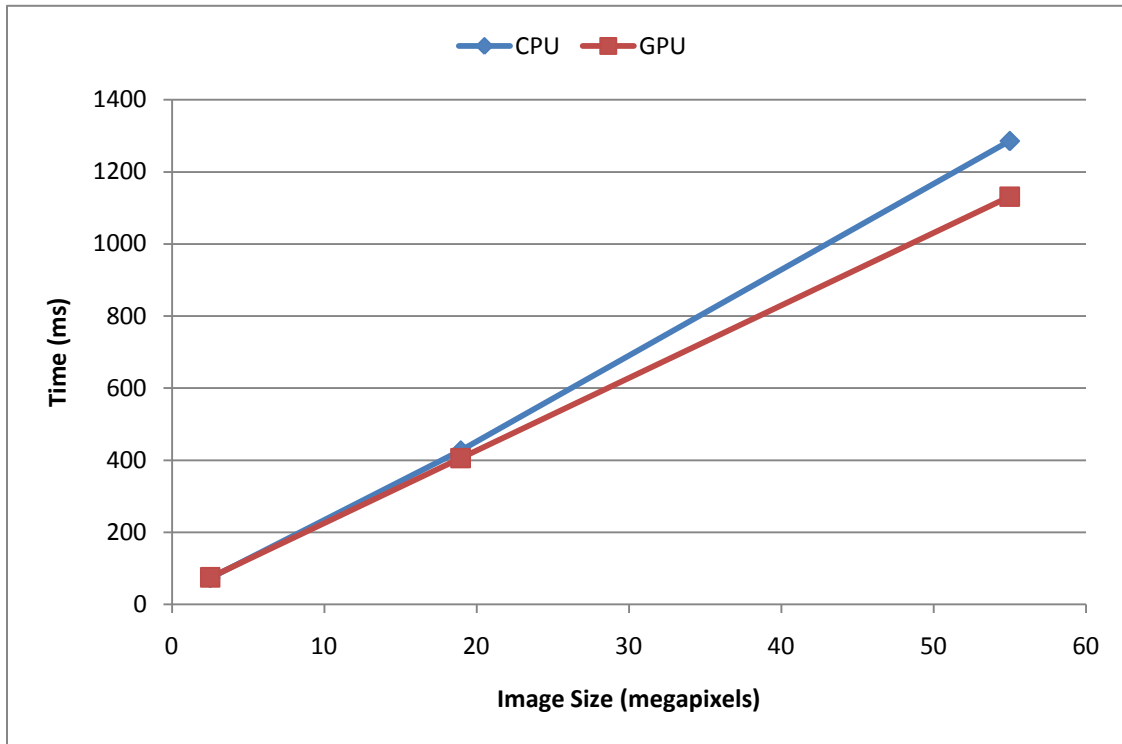


Figure 3 – Comparison of time to crop images

It is likely possible to convert the un-optimized portions of the crop operation to parallel GPU based implementations, but that task is reserved for future work. The step to compute the bounding box is particularly attractive because it must iterate through the entire image and all indications are that this portion is primarily responsible for the majority of remaining time. Parallelizing this portion should result in vastly decreased execution time for the crop operation on the GPU.

6. Sharpening Operation Results

Images are sharpened by FamilySearch to improve the contrast of the written portion of the document against the paper it was written on to make it more readable. The sharpen operation used by FamilySearch utilizes the common Unsharp Mask algorithm that can be decomposed into three steps:

1. Perform a Gaussian Blur on the source image
2. Take the difference of the blurred image from the original and multiply it by a specified amount

3. Add the image produced from the previous step and clamp any values back to the displayable range of [0,255]

As with the crop operation, the NPP library provides suitable methods to increase performance through parallelization on the GPU. The Gaussian Blur in Step 1 using NPP's `nppiFilter` function was practically instantaneous on the GPU, not even registering a millisecond of execution time in performance measurements. In contrast, the IPP `ippiFilter` operation took between 100-200 ms depending on the image size.

A much larger portion of the computation time for a CPU based sharpening operation takes place in Steps 2 and 3 because the code must iterate through the entire images sequentially to subtract, multiply and add pixel values. Unlike the portions of the crop operation that were left on the CPU, the NPP library also has functions to perform these operations. Figure 4 shows how performing sharpening on the GPU provides a dramatic speedup of about 7x over the CPU implementation for all image sizes.

Image Name	Megapixels	CPU	GPU	Speedup
MARY0004D_Estate-Records-6456_06464-bl.tif	2.5	195.3	27.7	7.06
SAM0203_00013.tif	18.9	513.0	82.7	6.21
ITALRO02D_1-7-37_lsi.tif	55.0	1399.7	181.7	7.70

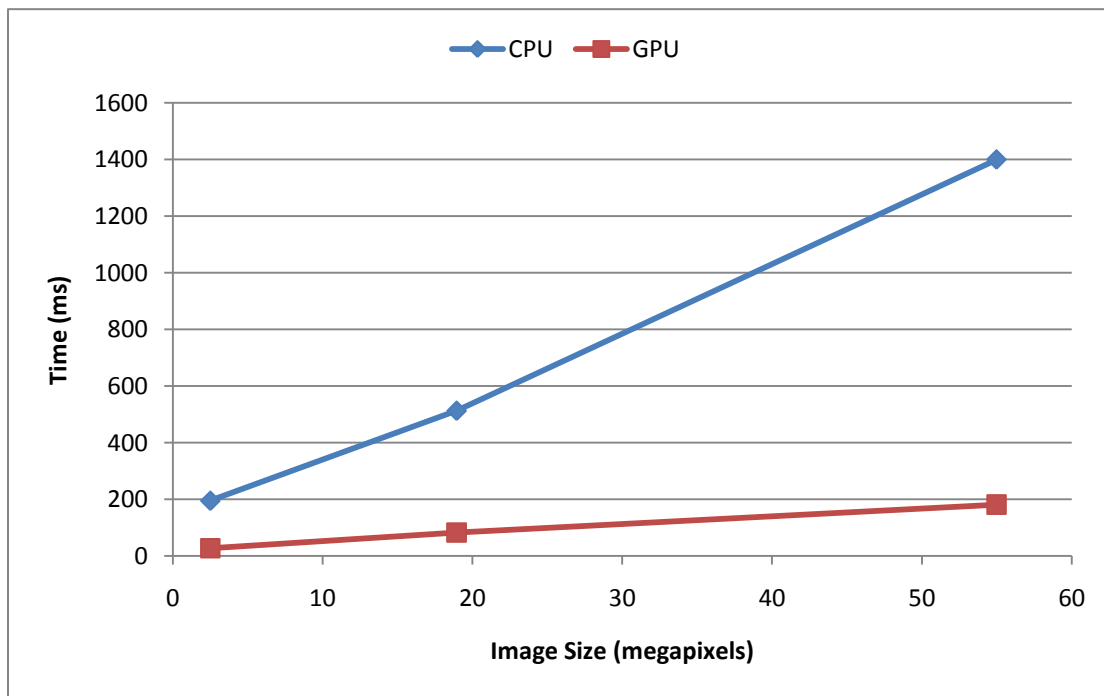


Figure 4 – Comparison of time to sharpen images

Ironically, this exercise helped discover that the current CPU based implementation could likely be improved by utilizing corresponding IPP functions for Steps 2 & 3. Doing so would provide a more fair comparison and probably improve performance on CPUs, though likely not as much as by utilizing the GPU.

7. Cropping and Sharpening Operations Combined Results

While it is instructive to examine individual image processing operations individually, operations in the DPC are not performed in isolation. Additional operations currently performed on each image include automatic skew correction, image scaling to produce thumbnail images and encoding/decoding in different image formats.

To begin to show what performance gains an entire DPC operational plan may have through parallelization on the GPU, the work on cropping and sharpening operations was combined without returning the image to the CPU between operations. Figure 5 shows a comparison of executing these two operations on each test image.

Image Name	Megapixels	CPU	GPU	Speedup
MARY0004D_Estate-Records-6456_06464-bl.tif	2.5	229.7	86.7	2.65
SAM0203_00013.tif	18.9	929.7	513.0	1.81
ITALRO02D_1-7-37_lsi.tif	55.0	2769.3	1460.0	1.90

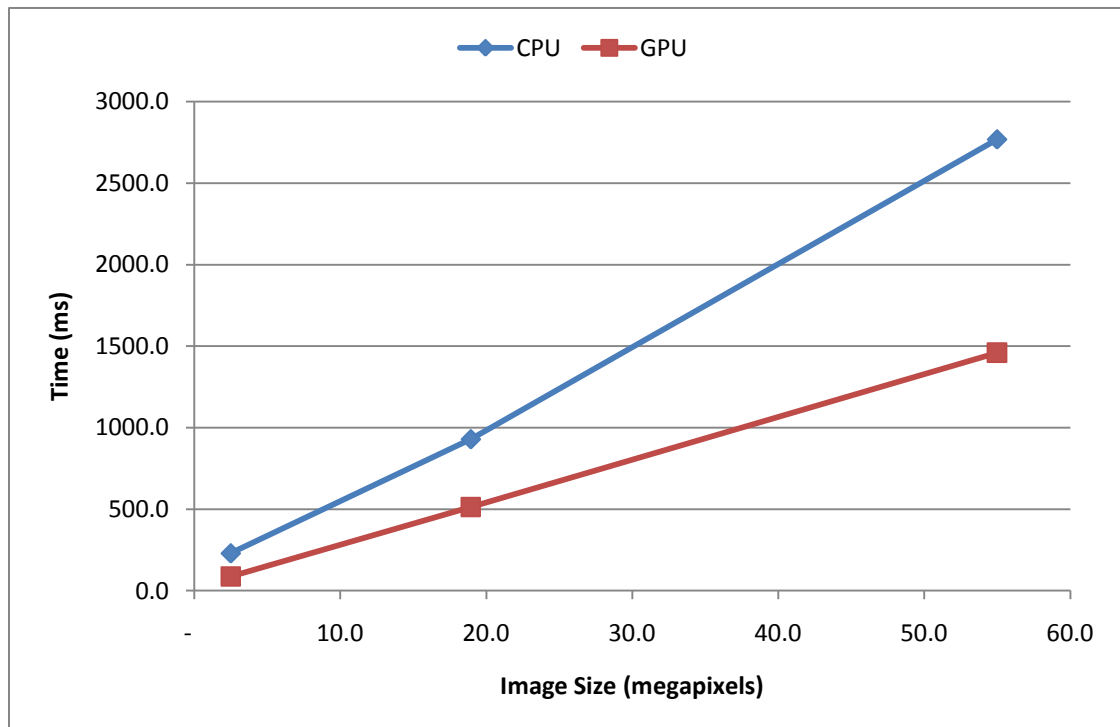


Figure 5 – Comparison of time to crop and sharpen images, including GPU transfer time

One item not previously discussed is that moving to a GPU based solution comes with additional costs in the form of transferring data to/from the GPU for computation. The previous results have not considered this time when comparing performance of operations, but this comparison does include GPU transfer time to illustrate the total expected performance of a GPU based solution. To maximize performance, as few transfers back and forth from the GPU as possible should be made.

Overall, the results of the cropping and sharpening operations in tandem provide roughly a 2x speed improvement on the GPU. This is especially significant considering the relatively un-optimized crop operation being a large part of the total time. If GPU optimized versions of all DPC operations were written it is expected that the overall speed improvement would at least meet and likely exceed the 2.5x speedup found by Lee et al. [15] when comparing various algorithms on CPUs and GPUs in a similar manner as done in this paper.

8. Conclusions and Future Work

This paper has shown that there is a significant increase in performance by parallelizing image processing operations for execution on GPUs. There also appears to be great potential in a GPU based approach to image processing at FamilySearch, especially if GPU performance continues to improve more quickly over time than CPU performance and the libraries, languages and tools for GPU computing continue to get better.

However, performance increases are highly dependent on the ability to take advantage of the data parallel nature of GPUs, but libraries such as NPP provide solutions that can be relatively easily applied to existing code bases such as the image processing library at FamilySearch without extensive knowledge of GPU programming.

Another potential improvement to image processing at FamilySearch by using GPUs is that more computationally intensive operations may be used to improve image quality without sacrificing current levels of performance.

In order to fully assess the viability of using GPUs for image processing at FamilySearch, the entire set of operations performed in the DPC should be implemented using GPU based libraries or languages to compare against the current CPU based library. This includes implementing or utilizing libraries to perform image encoding/decoding. One potential solution for JPEG-2000 encoding/decoding is the Cuj2k library [16].

To better validate actual throughput improvements, comparisons of CPU and GPU performance for image processing should also include an investigation in more production-like environments where thousands of images per hour are simultaneously processed across multiple servers executing multiple threads each.

Finally, as FamilySearch increases the number of images processed it is believed that the performance gains shown in this paper will result in decreased processing times per image, increased throughput, smaller form factor and decreased total cost of ownership.

9. References

1. *Using Graphics Devices in Reverse: GPU-Based Image Processing and Computer Vision.* James Fung, Steve Mann.

2. *A Survey of General-Purpose Computation on Graphics Hardware*. *Computer Graphics Forum*, 26(1):80–113, March 2007. John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Tim Purcell.
3. General-Purpose Computation on Graphics Processing Units. [Online] <http://gpgpu.org>.
4. *Languages, APIs and Development Tools for GPU Computing, 2010 GPU Technology Conference*. Ramey, Will.
5. CUDA Zone. [Online] http://www.nvidia.com/object/cuda_home_new.html.
6. CUDA Course Map - CUDA University Courses. [Online] http://www.nvidia.com/object/cuda_courses_and_map.html.
7. High Performance Computing (HPC) - Supercomputing with NVIDIA Tesla GPUs. [Online] http://www.nvidia.com/object/tesla_computing_solutions.html.
8. Intel® Integrated Performance Primitives Performance Library - Intel® Software Network. [Online] <http://software.intel.com/en-us/articles/intel-ipp/>.
9. Taylor, Stewart. *Intel Integrated Performance Primitives, How to Optimize Software Applications Using Intel IPP*.
10. *OpenCV Wiki*. [Online] <http://opencv.willowgarage.com/wiki>.
11. *NVIDIA NPP*. [Online] http://developer.nvidia.com/object/npp_home.html.
12. *NVIDIA Performance Primitives & Video Codecs on GPU, 2009 GPU Technology Conference*. Anton Obukhov, Frank Jargstorff.
13. *GPUCV: A Framework for Image Processing Acceleration with Graphics Processors*. Jean-Philippe Farrugia, Patrick Horain, Erwan Guehenneux, Yannick Alusse.
14. FamilySearch.org - Indexing. [Online] http://www.familysearch.org/eng/indexing/frameset_indexing.asp.
15. *Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU*. Victor W Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupati, Per Hammarlund, Ronak Singhal and Pradeep Dubey.
16. JPEG 2000 on CUDA. [Online] <http://cuj2k.sourceforge.net/index.html>.