Accelerated 2-D and 3-D Digital Image Processing on a GPU

Bryson R. Payne
Georgia College & State University
bryson.payne@gcsu.edu

G. Scott Owen, Saeid O. Belkasim, and Patrick Flynn
Georgia State University
sowen, sbelkasim@cs.gsu.edu; patrick@biltmorecomm.net

1 Introduction

Programmable graphics processing units (GPUs) are hardware commonly included in new computer workstations, including those workstations used for digital signal and image processing. The current generation of GPU is roughly equivalent in processing power to current CPUs (Central Processing Units), but that power is rarely used to its full capabilities in an average signal processing workstation. Numerous advances have been made recently in applying the GPU to both graphical and non-graphical parallel matrix processing tasks, such as the Fast Fourier Transform (FFT) [Moreland and Angel 2001; Wloka 2003], wavelet analysis [Wang et al. 2004] and many others.

Our goal was to research the implementation of common image and signal processing algorithms on the GPU versus the same algorithms on a CPU, with special attention to high-resolution image formats such as HDTV video.

2 Background

Research has been previously reported on the advantages of GPU computation over CPU computation with respect to convolution filters [Viola et al. 2003], but little research has addressed the application of these approaches to image processing at interactive frame rates at very high resolutions such as the HDTV standard format (1920x1080 pixels). The purpose of this research was to examine the GPU's capabilities in this area versus a comparable modern CPU. The GPU used in this research was a GeForce FX5900, 400 MHZ GPU with 256 MB of RAM. The CPU used for comparison was a 2.8 GHz Pentium 4 with 1 GB of RAM.

3 Implementation

We developed GPU and CPU versions of various image and signal processing filters for 2D images and video from disk and 3D-rendered video output from the GPU at image sizes from 64x64 to 4096x4096. Included were 3x3 averaging, Sobel edge detection, Gaussian smoothing and median filters, 3x3 binary morphological operators for erosion and dilation, an implementation of binary skeletonization, and 5x5 Gaussian smoothing.

In all cases, care was taken to implement the most efficient and equivalent algorithm for each processor (GPU and CPU), and special attention was paid to the portability of the code between Linux and Windows platforms. Fortunately, all source code was able to be developed for use on both platforms with no changes necessary.

4 Results

The GPU proved to be much faster than the CPU at applying convolutions like Gaussian smoothing, edge detection, and averaging filters at high resolutions (1024x1024 and higher). In the case of a 3x3 convolution filter, the GPU was up to 30 times faster than the CPU, even after communication between main memory and the GPU was accounted for. For 5x5 convolution filters, the GPU gained an even greater advantage, performing at 50-60 times faster than the CPU for images measuring 1024x1024 to 4096x4096, a range which includes HDTV-quality 1920x1080-pixel images. Again, this advantage includes the cost of transferring images to and from the GPU, so it applies to video on disk or DVD as well as video produced or captured on the graphics card.

Perhaps the most exciting result for display-based image processing was the fact that the GPU was able to maintain interactive frame rates (38 frames per second) at 1024x1024 resolution using a 3x3 averaging filter. The result was much better anti-aliasing (in the case of 3D rendering) than traditional 4-pass rendering for a negligible cost in terms of frame processing time. The CPU could only process 1.8 frames per second at that resolution. Even at 2048x2048 (roughly double HDTV-quality), frame rates higher than 15 fps were achieved on the GPU, showing that acceptable real-time results can be achieved on today's hardware for applications at the highest useable current resolutions.

5 Conclusions

Perhaps one of the most promising conclusions of this research is the discovery that HDTV-quality images and video can be processed by convolution filters at real-time, interactive video frame rates, using hardware already readily available in newer computer workstations. Current and future video-ready cards, which allow direct video input through the graphics card, could take advantage of this capability and apply various filters in real-time to HDTV-quality video or images from virtually any source. This opens the door to real-time HDTV-quality video processing and editing on consumer-level workstations for a fraction of the cost of current HDTV editing solutions.

In addition, building on the ability to perform convolutions, including Sobel edge detection and other valuable image processing tasks, further processing for applications such as computer vision and robotic navigation at extremely high resolutions (i.e. HDTV resolution) could be performed either on the GPU, the CPU, or a combination of the two. This high-speed, high-resolution processing could provide significant improvements in the interactive and navigational capacities of current and future robotics applications.

One final advantage of this research was the fact that the programs used were directly portable to both Windows and UNIX platforms with no changes needed. The standard OpenGL, GLUT, and Cg or OGSL interfaces allow the exact same code to be run on platforms from Windows XP to RedHat Fedora to Mac OS X, allowing for greater portability and collaboration among systems.

References

MORELAND, K. and ANGEL, E. 2003. The FFT on a GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware*, 112-119.

VIOLA, I., KANITSAR, A., and GRÖLLER, E. 2003. Hardware-Based Nonlinear Filtering and Segmentation using High-Level Shading Languages. Technical Report TR-186-2-03-07. Institute of Computer Graphics and Algorithms, Vienna University of Technology.

WANG, J., WONG, T.-T., HENG, P.-A. and LEUNG, C.-S. 2004. Discrete Wavelet Transform on GPU. World Wide Web site available online at http://www.cse.cuhk.edu.hk/~ttwong/software/dwtgpu/. May 2004.

WLOKA, M. M. 2003. Implementing a GPU-Efficient FFT. In *ACM SIGGRAPH 2003 Course Notes*, pages 132-137. ACM SIGGRAPH, August 2003.