CSCI1950V Project 5 : Real-time Video Filtering

April 4, 2012

1 Summary

In this project, you will use CUDA to quickly perform filtering operations on frames from a video stream. The filtering concepts are very similar to those in CS123's filter assignment. You will probably want to try implenting similar filter types and mathematical computations.

This project is due at 11:59PM on April 20, 2012.

2 Requirements

You must implement convolution filtering with variable width kernels, and provide a way to construct such kernels for at least one such filtering effect (blur, for example). You convolution must handle edge cases either by adjusting filter weights or repeating/extending the image. Darkening at the edges of the image with a blur, for example, is not okay.

Feel free to add UI elements for control over filter parameters, and we encourage you to use your filters for exciting postprocess effects (bloom, for example).

You must implement some sort of shared memory usage. You can only spawn up to 512 threads per block on our cards, and there are usually more than 512 pixels in the video frame, so you will need to spawn multiple thread blocks. If each thread block operates on a certain part of the destination image, you should only need a subset of the original image to run your filtering operations. The actual shared memory scheme and copying mechanics are up to you, as is your grid/block/thread structure.

Turn in with your code some data and charts about your application performance with different kernel sizes, grid/block sizes, and shared memory vs global memory usage. CUDA events can help you to benchmark performance of your application (see page 34, section 3.2.5.6.2 of the CUDA C programming guide).

3 Support Code

You can copy the support code from /course/cs195v/cudasupport. We provide an example kernel to invert an image. The code can play video files from a number of sources, including the internet. Use a website like keepvid.com to convert youtube links into readable links for the program.

4 Tips

- If using a separable filter, you should call cudaDeviceSynchronize() between filtering the rows and the columns to make sure that the first kernel finished executing before you start the second
- Similarly, you should call __syncthreads() in between copying shared memory and using it
- To add a new .cu file to the qt project, you should add it to the CUDA_SOURCES and OTHER_FILES variables in the qt project file, not SOURCES, since .cu files must be compiled separately by nvcc