Assignment 3 Report

CMPT – 431

Joshua Campbell - jkcampbe@sfu.ca

Adam Penner – adpenner@sfu.ca

**Analysis:**

On the server (which appears to be running a GTX 760 (Kepler architecture), we can see that both the Tiled and Naïve implementations of the program are faster than the serial CPU version of the code for all of the image processing techniques. However, the naïve code tends to be slightly faster than the tiled implementation of the code (around a 4-5 millisecond difference). The main area where these two implementations occur is in the calculation of the histogram (the gpu\_histogram cuda kernel function in histogram-equalization.cu). This is the only place that we could find where the structure of the program would possibly allow for a benefit from using shared memory (tiling). However, due to the nature of how the histogram is calculated, you must use atomicAdds in order to prevent race conditions between the threads if they happen to be modifying the same address at the same time in order to get a correct histogram (and eventually a correctly processed image). As we can see from these charts, this use of shared memory provided no performance gains on the server. However, when the same code was run on a GTX 675m (NVIDIA’s Fermi architecture), the tiled code executed quite a bit faster (around twice as fast when doing a grayscale image processing) than the naïve implementation (diagrams can be found below). After doing some research into the matter, the NVIDIA Kepler Architecture Whitepaper described some improvements that were made from the Fermi to the Kepler architecture that could explain this huge difference in performance gains. One of the improvements in Kepler described in the whitepaper is that atomic operations such as atomicAdd have received a 9 times performance increase from the previous Fermi architecture (page 12 in the included NVIDIA document).