CSC367 Assignment 4 Report

How to run the code:

- bash collect_data.sh

All tests were performed on lab machines

CPU implementation:

In our first part of the assignment, we had to use CPU implementation from A2 for image processing. From A2 we have discovered that the most efficient and fastest CPU implementation was work queue, so we decided to use it and compare it to our future GPU kernel implementations.

GPU implementations:

Our first GPU implementation was kernel1, 1 pixel per thread, column major. Column major was done by identifying the index of kernel (int idx = blockIdx.x*blockDim.x + threadIdx.x) and correct column and row (int col = idx%height; int row = idx/height) for the input matrix.

Kernel2 implementation with 1 pixel per thread, row major was done similarly to kernel1 implementation. However, since this time we work on rows, columns and rows were found using width of the input matrix (int row = idx/width; int col = idx%width).

Kernel3 implementation has multiple pixels per thread, consecutive rows, column major. It was done by defining row (int row = blockIdx.x*blockDim.x + threadIdx.x). Hence, each thread identified by idx does its own row up to height – 1.

Kernel4 implementation has multiple pixels per thread, sequential access with a stride equal to the number of threads. Idx was found using int idx = blockIdx.x*blockDim.x + threadIdx.x;, int row = idx/width;, int col = idx%width; Hence, each thread identified by idx does its own row up to the height – 1.

Please note that since threads work on their own parts, no reduction is needed in any of the pixel computations.

It was also decided to use 512 threads per block and the number of blocks is determined by width*height + 512 – 1/ 512 where width and height are coming from input matrix. That particular number of blocks and threads was chosen in order to achieve optimal GPU memory bandwidth efficiency.

Since input and output matrices are 1-Dimentional arrays, accesses to sequential memory locations by threads are very fast.

Please check “output/” folder for graphs with performance of CPU and GPU implementations.