Cameron Calv and Nicholas Sica
ECEC 413 Intro to Parallel Computer Architecture
Assignment 4: Blur Filter (Cuda)

Parallelization

The code was parallelized by using each thread ID as a way to determine where in the image the program is going to be averaging. Each thread then takes that average and adds to the output what that value is. In this way, the image is split up into smaller pieces with each piece or neighborhood given to a thread to work with.

Evaluating Parallelization

Below are tables outlining the runtimes of the program run for parallelization. Following the tables are a plot showing the difference of the execution times for all data sizes.

Table 1: Evaluating the execution time of the program on Drexel COE college's Xunil server.

	Data Size - Blur Filter				
	512	1024	2048	4096	8192
Sequential	0.011223	0.041299	0.129705	0.454584	1.659496
GPU	0.145982	0.143968	0.157078	0.224967	0.373532

Table 2: Evaluating the speed-up of the program with parallel threads on Drexel COE college's Xunil server.

	Data Size - Blur Filter				
	512	1024	2048	4096	8192
GPU	-1200.74%	-248.60%	-21.10%	50.51%	77.49%

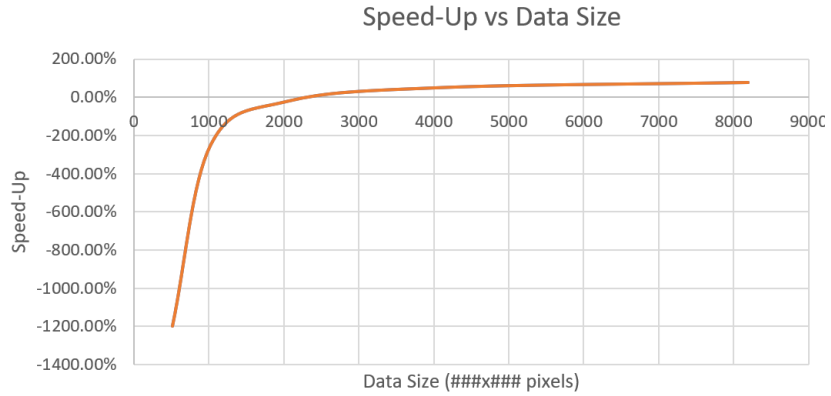

Figure 1: Plot showing the speedup versus the data size of the blurred image.

An increase in speed up does not occur until the image goes above a certain threshold. As per usual this is a result of various factors that more or less sum up to the larger the data, the larger the potential benefit of parallelization.