Analysis of the Execution Time Variation of OpenMp with Canny Edge Detection and Feature Detector

Marco Lin
University of the Pacific

Hight Performance Computing
m_lin23@u.pacific.edu

***Abstract*—The purpose of this report is to compare the difference of execution time between serial execution and parallel execution. In order to analyze this, we will implement Canny Edge Detection and Feature Detector first and, and then using open MP to parallelize it.**

Keywords—C, Canny Edge Detector, open MP, parallel

I. Introduction

The purpose of this report is to analyze the speedup and parallel efficiency by write a serial and parallel code. In Canny Edge Detector and feature detector, we have to implement convolution, Suppressed, and Hysteresis. In those function, the program needs to get and calculate the value of each pixels. This process take up most all of execute time thus by using to parallelize it, we can effectively to observe the the changing of execution time. In addition, running our program on cluster which provide multiple processors allows us to observe the changing of execution time between different number of threads. (From two threads to thirty-two threads)

II. Implementation

A. Canny edge detector

We used this function to detect the edge of image. The outputs will show below.

Figure 1.

Figure 2.

Figure 3.

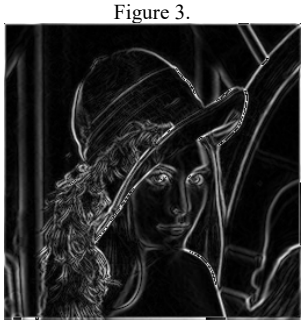

Figure 4.

Figure 5.

Figure 6.

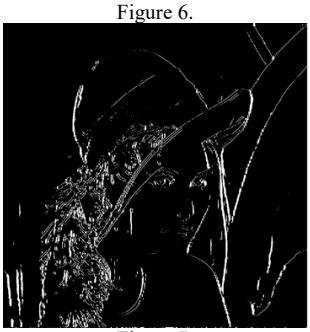

Figure 7.

The Figure1 is the original image we input to program. The pixels of image were calculated with 1D Gaussian kernel first. The Figure2 and the Figure3 are the output of horizontal gradian and vertical gradian. Then we used horizontal gradian and vertical gradian to calculate the gradian phrase. The Figure4 and the Figure5 are magnitude and Direction. In order to identify the edge of image cleaner, we need to do Suppression Threshold and Hysteresis function as well. The Figure6 is the output. At the end, by the values from previous function, the Figure7 is the output which shows the edge of image.

B. Parallelization Methodology

By setting two time zones: end-to-end time and computation time to measure the execution time in serial code and parallel code. we parallelized the for loop most because for loops take up most of all execution time. The environment we used to run the program is a cluster which allows us to use multiple processors. The sizes of images we use were: 12800, 10240, 7680, 4096, 2048, to 1024. And the value of sigmas were 1.25, 1.1, 0.6. Moreover, the threads we test were 2, 4, 8, 16, 32.

III. Results and Analysis

We have the figures from 8 to 13 to show our Speed-up time and Efficiency results in each sigma.

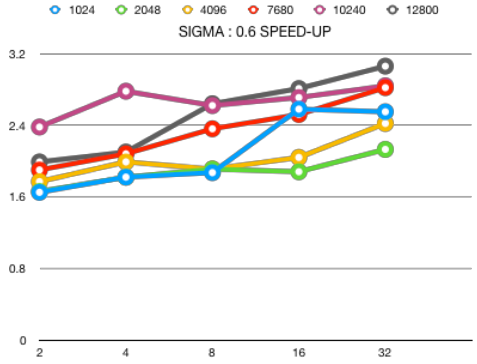

Figure 8

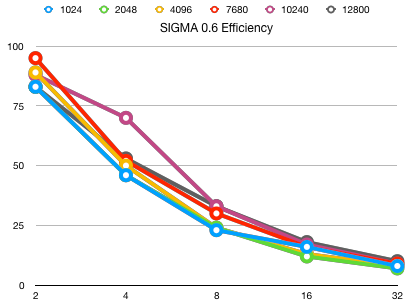

Figure 9.

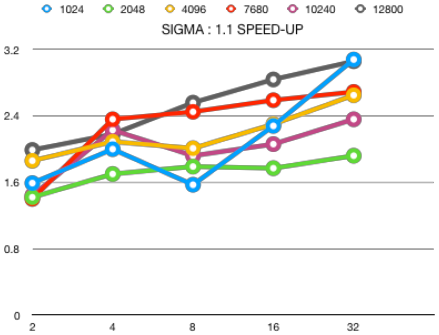

Figure 10.

Figure 11.

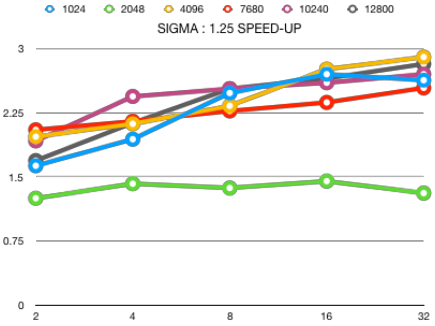

Figure 12.

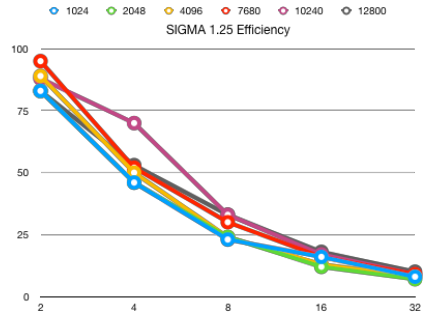

Figure 13.

The execution time are affected by sigma value, size of image and number of threads. According to the execution time we got, we show the Speed-up time and Efficiency. The average of speed-up time is around 2.6. At the first time, I put too much parallel command to parallelize the for loop. Then, I found the execution time didn't decrease. Instead, it was increasing. I only parallelized the convolve function at the final. So, it appears that more parallelization is not along with less execution time.

IV. Conclusion

Along with this project, we could understand the execution time will affected by not only the condition we set but also the condition of cluster. My serial code on the first time running, it showed me the execution time of serial code was faster than parallel code. Then after I have run more times, the serial code was slower than first time. According to our results, the open MP can speed-up the program 2.6 times.