CSC2002S

Assignment 1

1. Parallelization Algorithms:

The basic aim of this assignment is to get exposure to parallel programming using the Java Fork-Join, the parallel program will be benchmarked to determine the efficiency, and hence if parallel programs are indeed worth the headache.

a) MeanFilterParallel

The first thing you do is to extract pixel values from a given image. The program takes as input an image that has a ‘jpeg’ extension. The pixel values have to be in a particular order to be able to return to their original place once the values themselves have changed, once ordered they are pixel values populated in an array.

The fork/join framework uses a ‘divide and conquer approach where the array containing the pixel values is basically divided and conquered recursively until a certain threshold is met. Then the ARGB values for each pixel are extracted and summed up into one value that is within the specified sliding square window. This value is divided by the value of the square window size to determine the mean alpha, mean red, mean green and mean blue values.
The left and right arrays are combined into a single array that contains the newly calculated values and these values are written into a new image file.

b) MedianFilterParallel

This class also implements the fork/join framework whereby an array is broken down recursively using the divide and conquer method. The ARGB values for each pixel are extracted from the input image and then sorted in ascending

order. The extracted pixel is then replaced with the middle value and stored in a new array with the newly calculated values. The results are written into a new image file.

c) Validation

In order to validate whether the parallel programs produce the exact same results as their serial counterparts.
i) The image size and quality must be the same for both the parallel and serial counterparts.
ii)The output images of both serial and parallel solutions will be visually examined to determine the accuracy and verify that the is a difference in quality for mean and median algorithms.

d) Timing

System.currentTimeMillis() method was used to time the algorithms. The start of the algorithm is when it receives the input image, so I created a System time variable called startTime and invoked the currentTimeMillis() method, and then created and invoked another System time variable at the end. I calculated the difference between these two and that was the execution time of the algorithm.

```
//Smoothing the image with sliding window approach
startTime = System.currentTimeMillis();
MeanSmoothFilter(height,width,bImage,finalImage,squareSize);

double endTime = System.currentTimeMillis();

double executionTime =  endTime - startTime;
System.out.println("\nTime taken = " + executionTime);
ImageIO.write(finalImage,  formatName: "jpeg", new File( pathname: arrString[1]+".jpg"));
```

e) Machine Architechures

- Lenovo V130-15IKB (2 cores)
- Ishango labs at the Computer Science Building - (4 cores)

f) Optimal Serial Threshold

The serial treshold value was determined by trial and error, and it it used to reduce the amount of serial computation.

g) Problems and Difficulties

The biggest problem I had was coming up with the algorithm for both the mean and median filter. At the start, the output image would just simply print out a black screen and for the longest time, I couldn't pip point out what the actual problem was. Another problem that I encountered was trying to implement the fork/join algorithm, and more so trying to generalize it in order to work for both the parallel programs, it was practically impossible so I used two different fork/join implementations.

2. Results

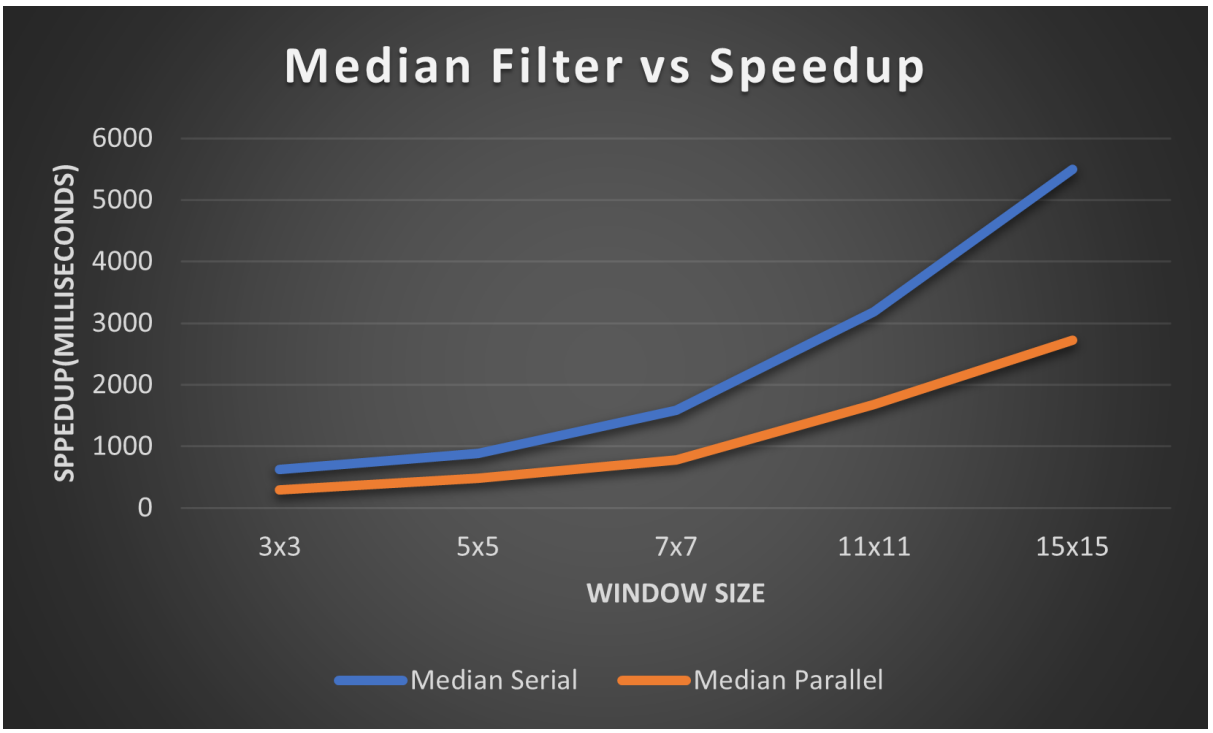
The data will be tested on two machines and in each machine both the mean and median algorithms will be tested, with their respective serial and parallel counterparts.

They will be tested using the following window sizes:

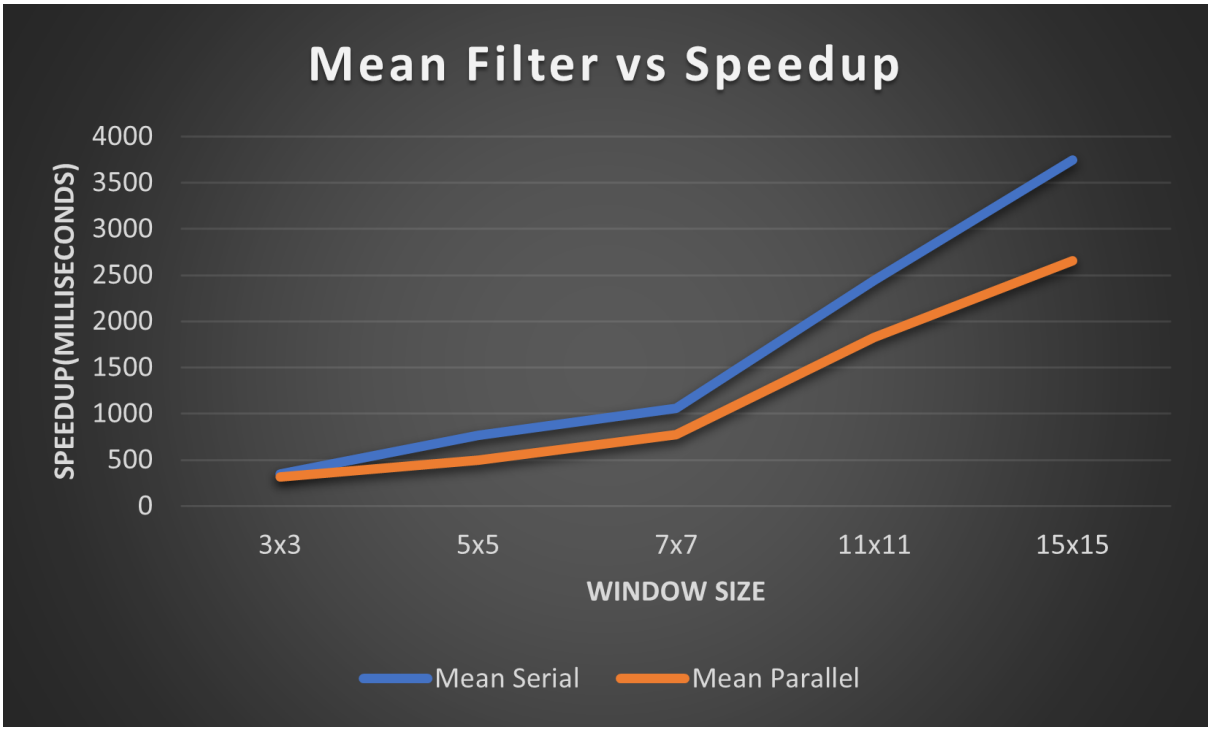
- 3x3
- 5x5
- 11x11
- 13x13
- 15x15

Lenovo 130-15IKB

1. Median Filter

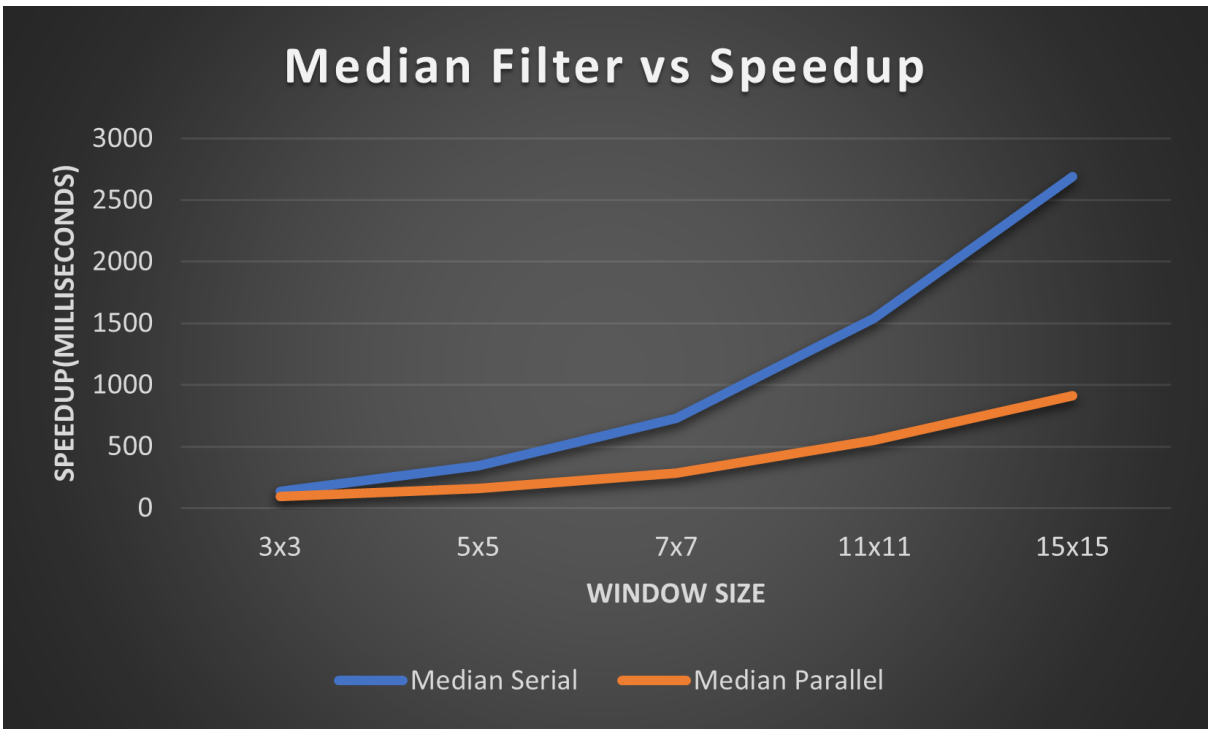


2. Mean Filter

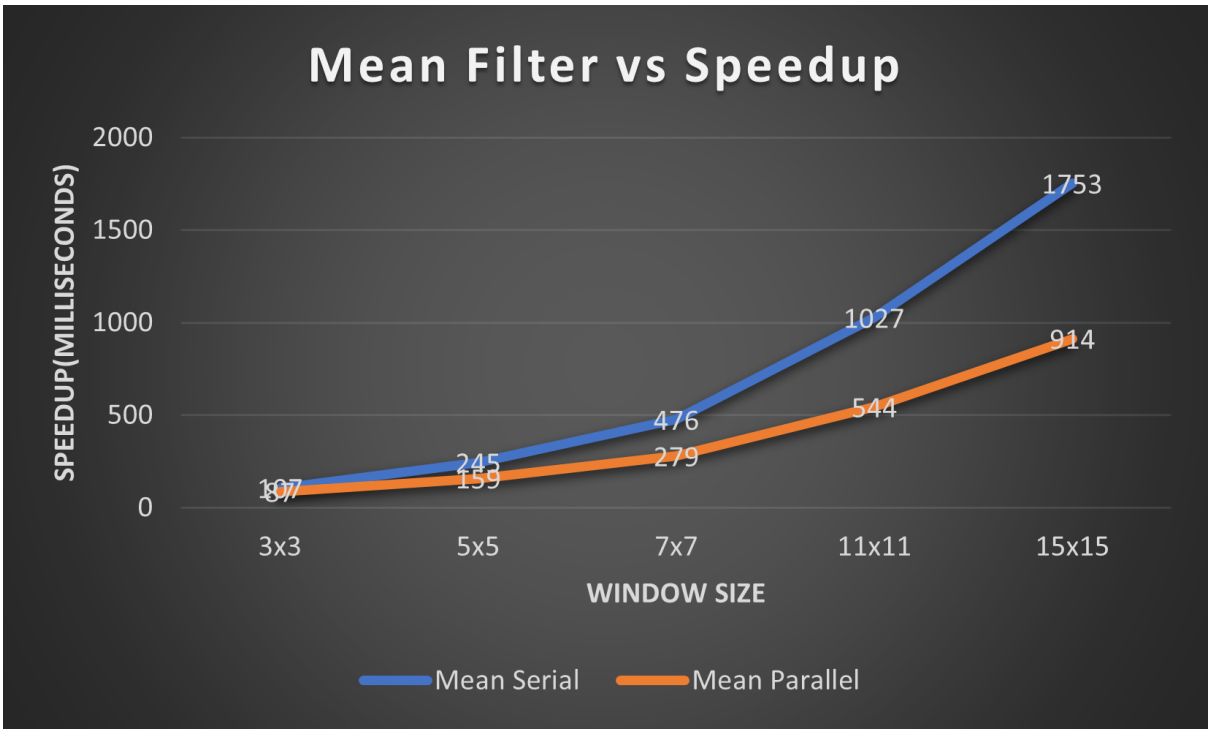


Samsung si-dual-170 (Comp Sci labs)

1. Median Filter



2. Mean Filter



3. Discussion

It seems smaller sequential cutoffs benefit and accommodates most window sizes as can be deduced from the graphs, hence smaller sequential cutoff values are optimal for both parallel algorithms.
My parallel programs are very consistent with smaller filter sizes as displayed on the graph.
The mean algorithms are relatively faster than their respective median programs, which makes sense because the process of obtaining a median value for each pixel is more complicated and tedious.
The parallel algorithms are also considerably faster than their serial counterparts, which was the point of the assignment.
Hence the aim of the assignment has been achieved.

I've also realized that the machine with the most cores is multiple folds faster than the machine with fewer cores. The time for execution for the iSHANGO labs computers was significantly less than that of my Lenovo pc.

4. Conclusion

The conclusion for this assignment is that a parallel program is indeed a fundamental part of programming, albeit difficult to implement, and can improve the efficiency of a program if utilized the correct way. It saves a lot of time most especially for programs when execution time plays a crucial role.