MSTISR001
STA2007H
Project 2

Sequential vs Multithreaded Image Smoothing

Abstract

This experiment tests the efficiency of both sequential and multithreading in image smoothing algorithms, making use of two methods: mean and median filters for comparison. The choice of employing two distinct algorithms, mean and median filters, allows for a better statistical analysis, considering their differing computational behaviours. Mean filters emphasise overall intensity and are computationally less complex, while median filters are more computationally intensive. The statistical analysis encompasses ANOVA, Welch's t-test, and Residual Analysis, providing a thorough evaluation of the data and supporting meaningful conclusions regarding the efficiency of multithreading in image smoothing algorithms.

Introduction

In the world of image processing, optimising the algorithm for performance is becoming more crucial by the day. One approach involves utilising parallel processing techniques, such as multithreading, to enhance computational efficiency. Multithreading allows for the concurrent execution of multiple threads within a program, thus improving the performance of a program while using more computational power. This experiment delves into the efficiencies gained through multithreading in the execution time of image smoothing algorithms. This will be carried out by a comparative analysis of image smoothing algorithms, specifically the median and mean filters, under both sequential and multithreaded implementations.

Methods

Programming Implementations:

Sequential Implementation:

In the sequential implementation, computations are executed in a linear, step-by-step fashion. Each task is completed before moving on to the next, resulting in a predictable and deterministic program flow. It is designed for computers with a minimum of one processor core that will run the program step-by-step until it is completed.

Multithreaded (Parallel) Implementation:

The multithreaded (parallel) implementation utilises concurrent execution, dividing tasks into smaller threads that can run simultaneously on multiple processor cores. This means that there needs to be a minimum of at least two processor cores on the computer to be able to divide the work among both cores, allowing simultaneous execution. Multithreading capitalises on modern multicore processors to handle multiple tasks concurrently, thereby potentially using more computational power.

Image Smoothing Algorithms:

A square grid(filter) traverses through the image to obtain the values(RGB) for each pixel(in the grid). A grids size can be set manually, the bigger the grid size the more pixels it covers which results in higher computational complexity. The pixel of interest for each instance of the grid will be the centre pixel, it will be the pixel that is altered. Due to this a grid width and length must an odd number. For this experiment a constant grid of 15x15 will be used in all algorithms for all images.

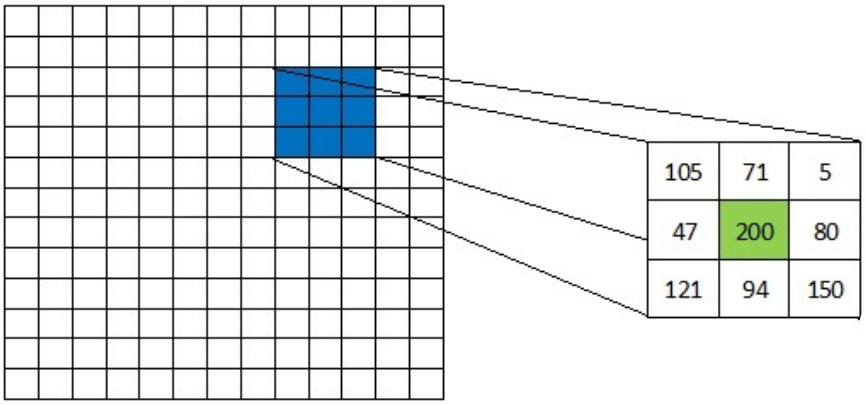

Fig 1.1

Fig 1.1 is the simplified representation of a 3x3 grid, the pixel of interest is highlighted in green.

Mean Filter:

In this algorithm, at an instance, the average of all the values in the grid is calculated and the pixel of interest is altered to this new value. The grid then continues to traverse through the image and repeats this process until the whole image is complete.

Fig 1.2

Fig 1.3

Fig 1.2 shows the original noisy image and **Fig 1.3** is the image after running the mean filter algorithm.

Median Filter:

This algorithm first takes the contents of a grid, at an instance, and sorts them in ascending order. The pixel of interest is then set to the median of this list. The grid then continues to traverse through the image and repeats this process until the whole image is complete. The addition of sorting causes this algorithm to be a lot more computationally complex than the mean filter algorithm.

Fig 1.4

Fig 1.5

Fig 1.4 shows the original noisy image and **Fig 1.5** is the image after running the median filter algorithm.

Experiment

Data Collection:

To carry out this experiment, the images, both input and output are hardcoded into the program along with the filter size(15x15). A timer in the program starts the moment the pixels of the image are being read and stops the instance the smoothed image is completed. This is done to ensure that there is absolutely no human error when recording times as well as to accurately record the time taken only for image processing and nothing else. The execution time is then printed out to the console to be recorded. All algorithms are implemented in the exact same way and recordings will be taken in the same manner for each algorithm for a number of varying computational work. An average time from ten executions from one algorithm on the same computational work will count as a single replicate e.g.:

MeanFilterSerial will be run 10 times with the same image, the average will be calculated and that will count as a single replicate.

The recorded data is then written to a csv file containing CWork, Algo, Type and ExTime.

Data	Description
CWork	The computational work(Total no. of pixels)
Algo	The type of smoothing algorithm(Mean filter, Median Filter)
Type	The type of program used(Sequential, Parallel)
ExTime	The execution time for image processing(Measured in Seconds)

Fig 2.1

Fig 2.1 displays the variables of interest in the experiment.

Observational Units:

The observational unit is the execution time of the various algorithms.

Treatment Factors & levels:

There are two treatment factors each having two levels:

Treatment 1: Serial Algorithms

- level 1: Mean Filter Sequential(MS)
- level 2: Median Filter Sequential(MDS)

Treatment 2: Multithreaded Algorithms

- level 1: Mean Filter Parallel(MP)
- level 2: Median Filter Parallel(MDP)

Experimental Units:

The experimental units in this experiment are the computational work, specifically the size of the images to be processed(e.g.: 3000x3000 jpg image = 9,000,000 pixels to be altered). Each image represents a distinct experimental unit upon which the image smoothing algorithms are applied.

Randomisation:

Images of the same size will share the same computational work(same amount of pixels). Due to this, four images of the same size will be labelled A, B , C and D, while the algorithms will be labelled MS, MDS, MP and MDP. The order in which the algorithms will be assigned to pictures for a specific size will be randomised in R. This is repeated before each set of replicates to avoid the possibility of one of the four algorithms replicating the same image size more than once, e.g.:

A	B	C	D
MDS	MP	MDP	MS

Pilot:

The pilot study comprised 8 replicates, during which the algorithms were randomly assigned to process images of two different sizes: 1000x1000 and 3000x3000. Specifically, the algorithms were randomly assigned to 4 images of size 1000x1000 and another set of 4 images of size 3000x3000. For each image size, the algorithms were applied, and the resulting execution times were averaged then recorded.

The primary objective of this pilot study was to determine the necessary sample size for the main experiment to achieve a power level of 87% with a 5% significance level. Cohen's d was utilised to quantify the magnitude of the difference in execution times between the two groups (sequential and parallel), aiding in the calculation of the effect size essential for the power analysis. 'n' in **Fig 2.1** represents the number of replicates for each group, so a sample size of 60 is needed to conduct the experiment.

power	significance	d	n
0.87	0.05	0.81	30

Fig 2.2

Fig 2.2 displays the output of the two-sample t test power calculation in R.

Model:

The model being examined: **ExTime ~ Type**

ExTime is the response variable (execution time).

Type is the explanatory variable (program type: Sequential or Parallel).

For H_0: $\mu_0 = \mu_1$

We expect there to be no significant difference in means between execution times in sequential(μ_0) and parallel(μ_1) programs.

For H_a: $\mu_0 \neq \mu_1$

We expect there to be some difference in means of execution times in sequential(μ_0) and parallel(μ_1) programs.

Analysis:

The statistical analysis to be carried out include:

ANOVA (Analysis of Variance):

- ANOVA is a statistical technique used to compare means among three or more levels of a categorical variable.
- It assesses whether there are statistically significant differences in means across the groups.
- ANOVA partitions the total variation in the data into between-group variation and within-group variation.
- The test produces an F-statistic and a p-value, indicating whether there's evidence of significant differences between groups.

Welch's t-Test:

- Welch's t-test is a variant of the two-sample t-test used when the variances of the two groups being compared are not assumed to be equal (heteroscedasticity).
- It's a test that allows for unequal variances and unequal sample sizes between the groups being compared.
- The test calculates a t-statistic and a p-value, indicating whether there's a significant difference in means between the two groups.

Residual Analysis:

- Methods include residual plots, Q-Q plots, and statistical tests.
- Residual analysis assesses the differences between observed and predicted values.

Results

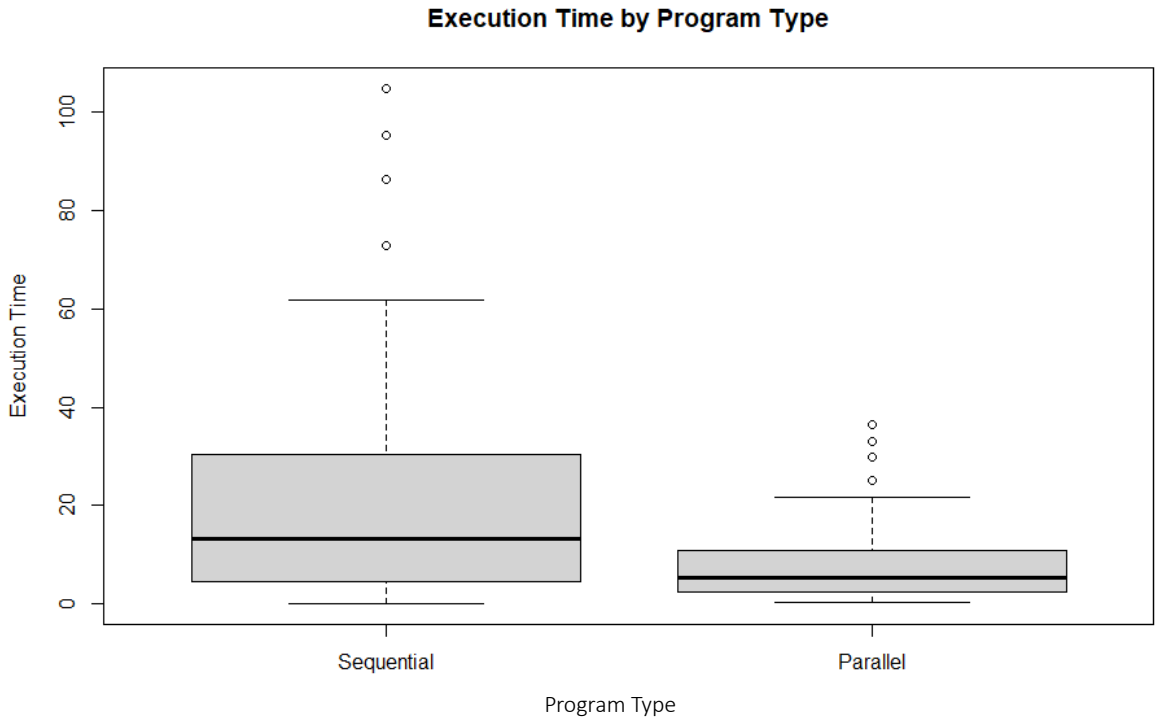

Fig 3.1

Fig 3.1 displays a boxplot of the execution time vs sequential and parallel program types.

The boxplot in **Fig 3.1** shows evidence that the values recorded for the parallel programs are consistently less than their sequential counterparts. Although, there is also evidence of outliers present in the data.

ANOVA:

An ANOVA model was created:

	Df	Sum Sq	Mean Sq	F-value	P-value
Type	1	3880	3880	7.666	0.00754
Residuals	58	293356	506		

Fig 3.2

Fig 3.2 displays a table representing R output of an ANOVA summary.

There is some evidence that with the given data, it is not likely that we would observe the null hypothesis due to the small p-value.

Residuals:

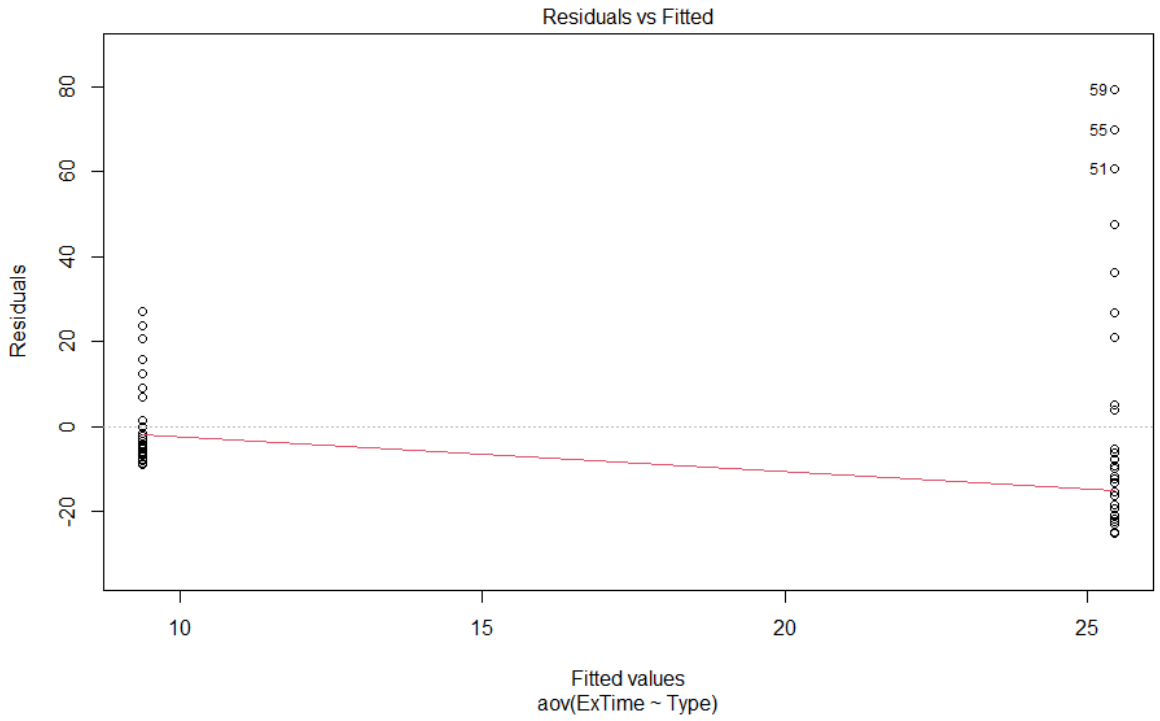

Fig 3.3

Fig 3.3 displays the residuals vs the fitted values of the ANOVA model.

There is evidence in **Fig 3.3** that the assumption of equal variance may be violated and using a Welch's t-test would be a better option than the ANOVA. This is evident by the points not being scattered randomly about the red line.

There is further evidence of this in the Scale-Location plot in **Fig 3.4**, there are no equally spaced points around the horizontal line.

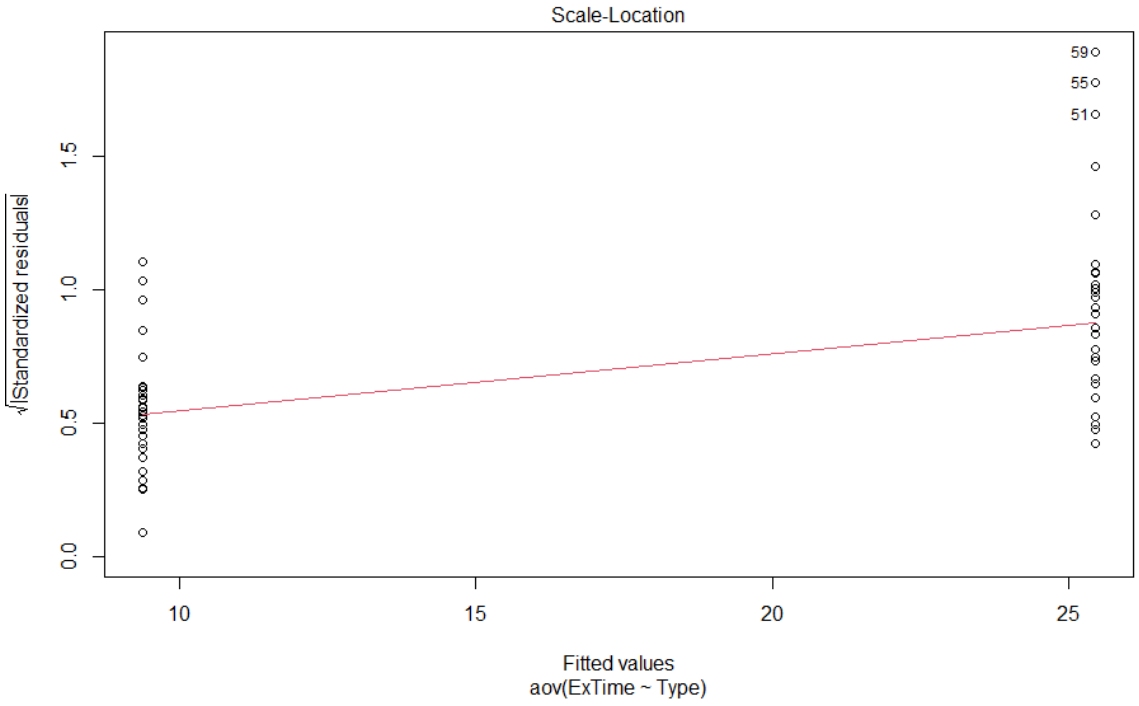

Fig 3.4

Fig 3.4 displays the scale-location plot for the ANOVA model.

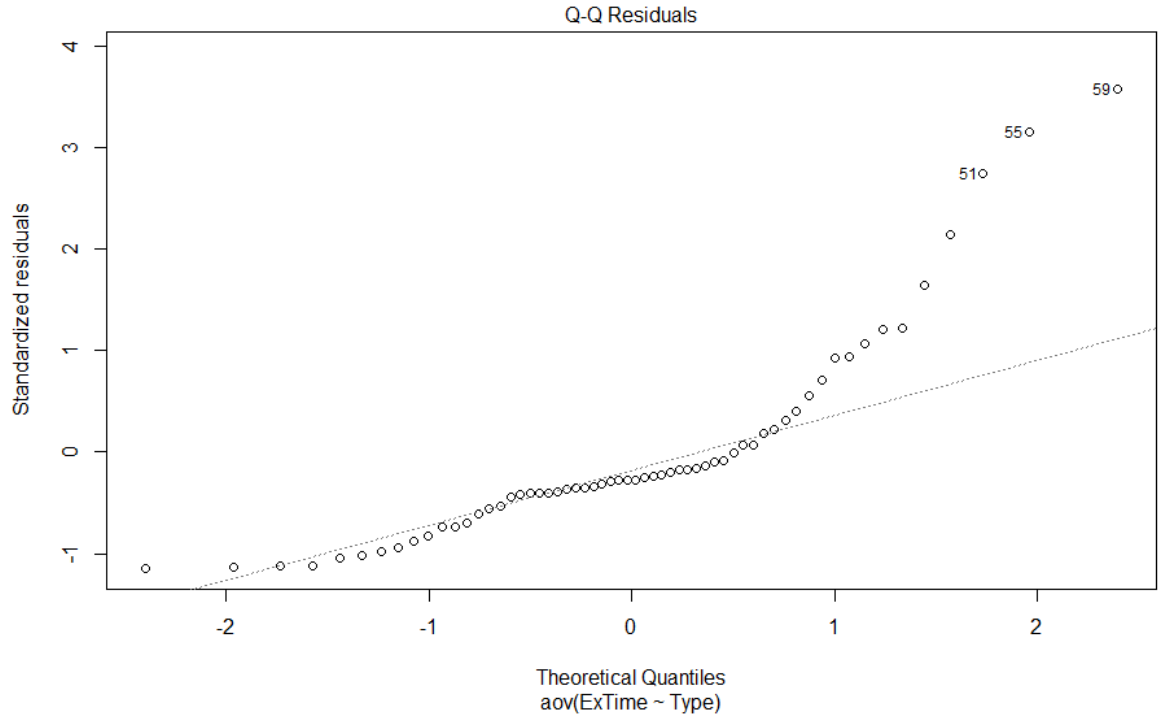

Fig 3.5

Fig 3.5 displays the Quantile-Quantile plot of the residuals.

The significant deviation from the line in **Fig 3.5** suggests that that the residuals are not normally distributed and are shown to be skewed to the right. More on this is available in the appendix.

There is also confirmation with high leverage within the Leverage plot in **Fig 3.6**.

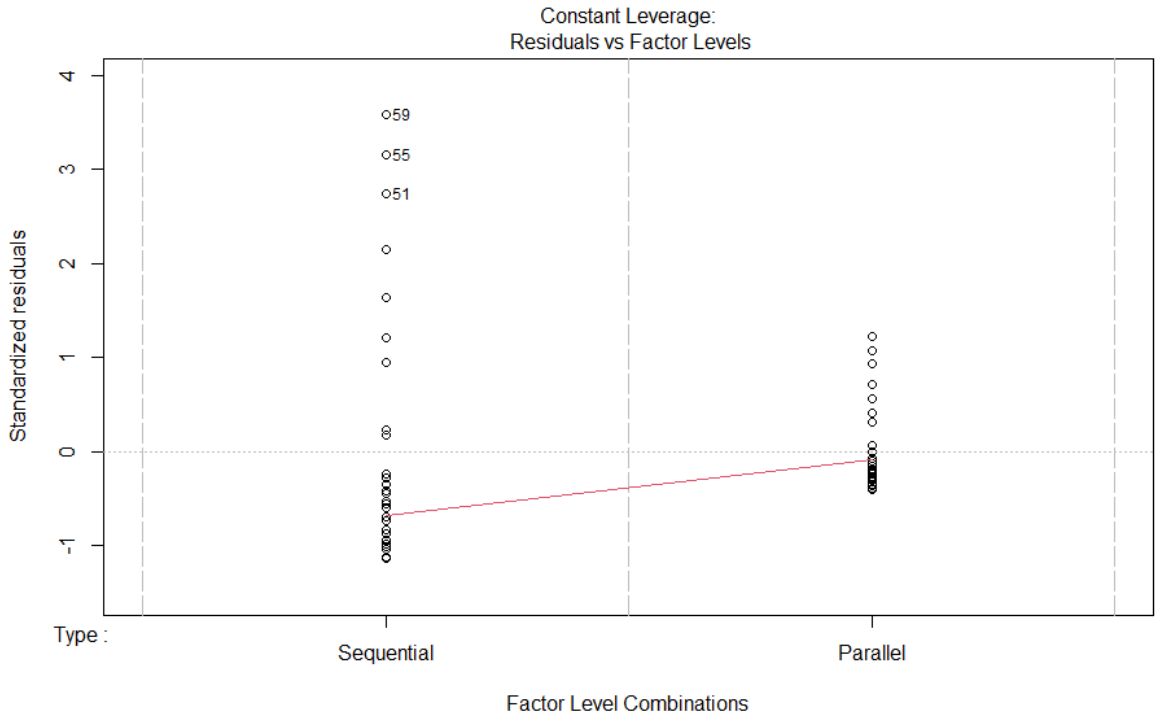

Fig 3.6

Fig 3.6 is the leverage plot for the ANOVA model.

Welch's T-Test:

The first 12 replicates were removed to get rid of the outliers, this came at a cost the power level of the experiment:

power	significance	d	n
0.793	0.05	0.81	24

Fig 3.7

Fig 3.8 displays the revised power analysis.

When running the Welch's T-Test it will take into account the possibility of unequal variance which is evident within the data.

t	df	p-value	95% conf.int	Est. mean: Sequential	Est. mean: Parallel
2.962	28.181	0.006	6.119, 33.530	31.269	11.445

Fig 3.8

Fig 3.8 is a table of the R output for the revised data's Welch's T-Test.

This confirms that there is very little possibility that this data could've occurred under the null hypothesis according to the test.

Conclusion

The experiment aimed to determine the impact of parallelisation on execution time in image smoothing algorithms compared to a sequential counterpart. By comparing sequential and multithreaded implementations of mean and median filters, the study found that parallelisation significantly reduces execution time. The results from the ANOVA test suggested a significant difference in means between sequential and parallel executions. However, this was not statistically substantial due to the outliers, equal variance assumptions being violated and the RSS not being normally distributed. After altercations and further analysis through Welch's T-Test, it was more appropriate due to potential unequal variances, revealing a significant difference in means with high confidence. The power analysis confirmed a high probability of detecting the true effect.

Appendix

Raw csv data:

CWork,Algo,Type,ExTime

40000,MeanFilter,Sequential,0.172

40000,MeanFilter,Parallel,0.335

40000,MedianFilter,Sequential,0.475

40000,MedianFilter,Parallel,0.328

160000,MeanFilter,Sequential,0.570

160000,MeanFilter,Parallel,0.524

160000,MedianFilter,Sequential,0.551

160000,MedianFilter,Parallel,0.723

1000000,MeanFilter,Sequential,2.431

1000000,MeanFilter,Parallel,1.337

1000000,MedianFilter,Sequential,9.134

1000000,MedianFilter,Parallel,3.380

1440000,MeanFilter,Sequential,3.024

1440000,MeanFilter,Parallel,1.689

1440000,MedianFilter,Sequential,13.161

1440000,MedianFilter,Parallel,4.837

1960000,MeanFilter,Sequential,3.733

1960000,MeanFilter,Parallel,1.749

1960000,MedianFilter,Sequential,15.601

1960000,MedianFilter,Parallel,5.390

2560000,MeanFilter,Sequential,4.596

2560000,MeanFilter,Parallel,2.450

2560000,MedianFilter,Sequential,20.065

2560000,MedianFilter,Parallel,7.594

3240000,MeanFilter,Sequential,6.101

3240000,MeanFilter,Parallel,2.849

3240000,MedianFilter,Sequential,29.419

3240000,MedianFilter,Parallel,10.792

4000000,MeanFilter,Sequential,7.081

4000000,MeanFilter,Parallel,3.250

4000000,MedianFilter,Sequential,30.459

4000000,MedianFilter,Parallel,10.813

4840000,MeanFilter,Sequential,9.129

4840000,MeanFilter,Parallel,3.954

4840000,MedianFilter,Sequential,46.310

4840000,MedianFilter,Parallel,16.271

5760000,MeanFilter,Sequential,10.107

5760000,MeanFilter,Parallel,4.341

5760000,MedianFilter,Sequential,52.084

5760000,MedianFilter,Parallel,18.400

6760000,MeanFilter,Sequential,12.082

6760000,MeanFilter,Parallel,5.433

6760000,MedianFilter,Sequential,61.788

6760000,MedianFilter,Parallel,21.749

7840000,MeanFilter,Sequential,13.602

7840000,MeanFilter,Parallel,5.782

7840000,MedianFilter,Sequential,72.906

7840000,MedianFilter,Parallel,25.205

9000000,MeanFilter,Sequential,16.120

9000000,MeanFilter,Parallel,6.375

9000000,MedianFilter,Sequential,86.172

9000000,MedianFilter,Parallel,29.914

10240000,MeanFilter,Sequential,17.590

10240000,MeanFilter,Parallel,7.120

10240000,MedianFilter,Sequential,95.317

10240000,MedianFilter,Parallel,33.087

11560000,MeanFilter,Sequential,19.358

11560000,MeanFilter,Parallel,9.201

11560000,MedianFilter,Sequential,104.660

11560000,MedianFilter,Parallel,36.432

Section 1:

Fig 1.6

Fig 1.7

Fig 1.8

Side by side comparison of different methods(**Fig 1.7** is Mean Filter, Fig 1.8 is Median Filter)

Section 2:

```
> #randomly assign
> ex <- rep(c("A", "B", "C", "D"), each = 1)
> print(ex)
[1] "A" "B" "C" "D"
> treats <- rep(c("MS", "MDS", "MP", "MDP"), each = 1)
> sample(treats)
[1] "MDP" "MDS" "MS"  "MP"
```

Fig 2.3, R code for randomisation per every computational work.

```
> serial <- c(2.431,9.134,16.120,86.172)  # Execution times for serial execution
> parallel <- c(1.337,3.380,6.375,13.004)   # Execution times for parallel executio
n
> # Calculate means and standard deviations for each group
> mean_serial <- mean(serial)
> mean_parallel <- mean(parallel)
> sd_serial <- sd(serial)
> sd_parallel <- sd(parallel)
> # Compute pooled standard deviation
> pooled_sd <- sqrt(((sd_serial^2 + sd_parallel^2) / 2))
> # Compute Cohen's d
> cohen_d <- (mean_serial - mean_parallel) / pooled_sd
```

```
>
>
> #power analyses for no. of replicates
> pwr.t.test(d = cohen_d, sig.level = 0.05, power = 0.87, type = "two.sample")

     Two-sample t test power calculation

              n = 30.07189
              d = 0.809413
      sig.level = 0.05
          power = 0.87
    alternative = two.sided

NOTE: n is number in *each* group
```

Fig 2.4, R code for calculation of no. of replicates using power analyses.

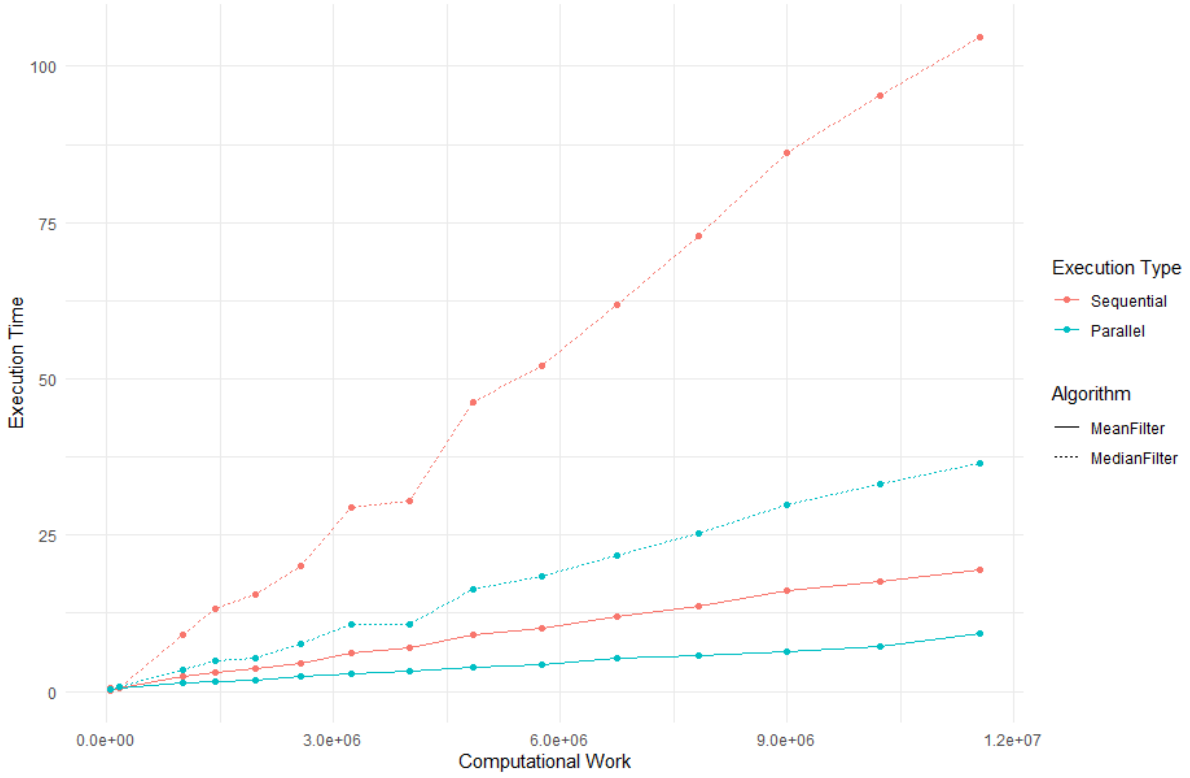

Fig 2.5, R plot of relationship between work and time for all algorithms.

Section 3:

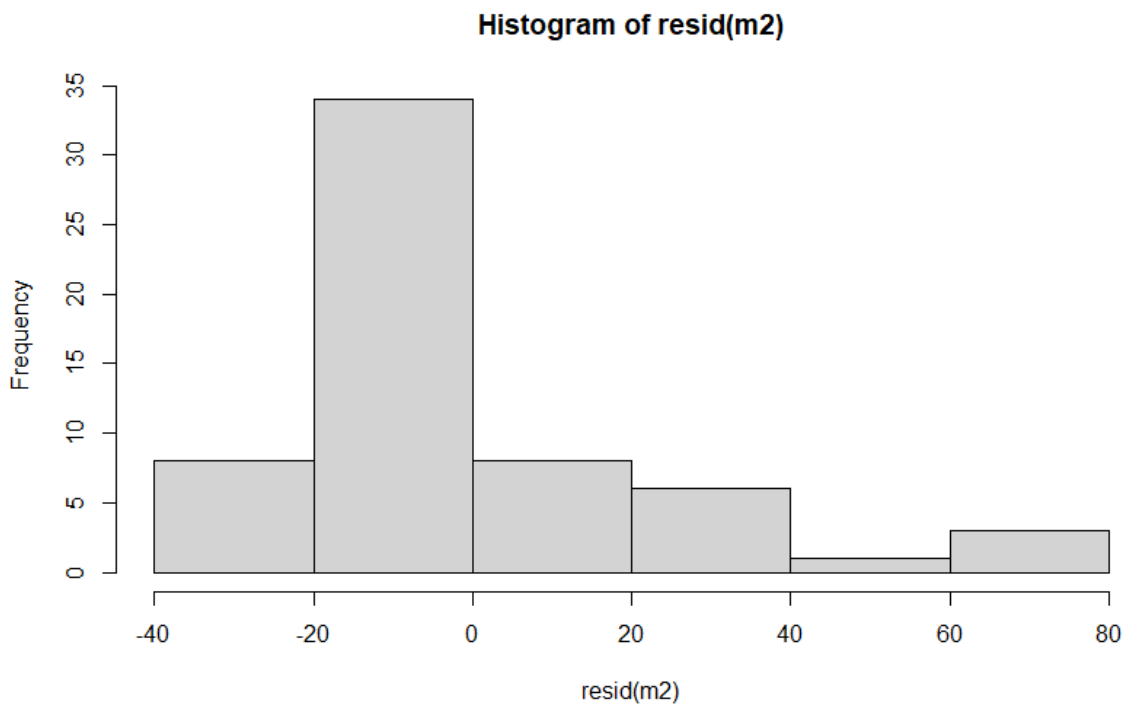

Fig 3.9, Histogram of residuals, showing confirmation of being right skewed.

```
> m2 <- aov(ExTime ~ Type, data = real)
> summary.aov(m2)
            Df Sum Sq Mean Sq F value  Pr(>F)
Type         1   3880    3880   7.666 0.00754 **
Residuals   58  29356     506
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> m2
Call:
   aov(formula = ExTime ~ Type, data = real)

Terms:
                    Type Residuals
Sum of Squares   3880.008 29356.035
Deg. of Freedom         1        58

Residual standard error: 22.49752
Estimated effects may be unbalanced
```

Fig 3.10, Initial ANOVA test with unaltered data

```
> pwr.t.test(d = cohen_d, sig.level = 0.05, power = 0.792, type = "two.sample")

     Two-sample t test power calculation

              n = 24.47819
              d = 0.809413
      sig.level = 0.05
          power = 0.792
    alternative = two.sided

NOTE: n is number in *each* group
```

```
> m1 <- t.test(ExTime ~ Type, data = real_subset)
> print(m1)

        Welch Two Sample t-test

data:  ExTime by Type
t = 2.9621, df = 28.181, p-value = 0.006145
alternative hypothesis: true difference in means between group Sequential and group
Parallel is not equal to 0
95 percent confidence interval:
  6.119093 33.529907
sample estimates:
mean in group Sequential   mean in group Parallel
                31.26938                 11.44487
```

Fig 3.11, revised power test and Welch's T-test after removal of outliers.

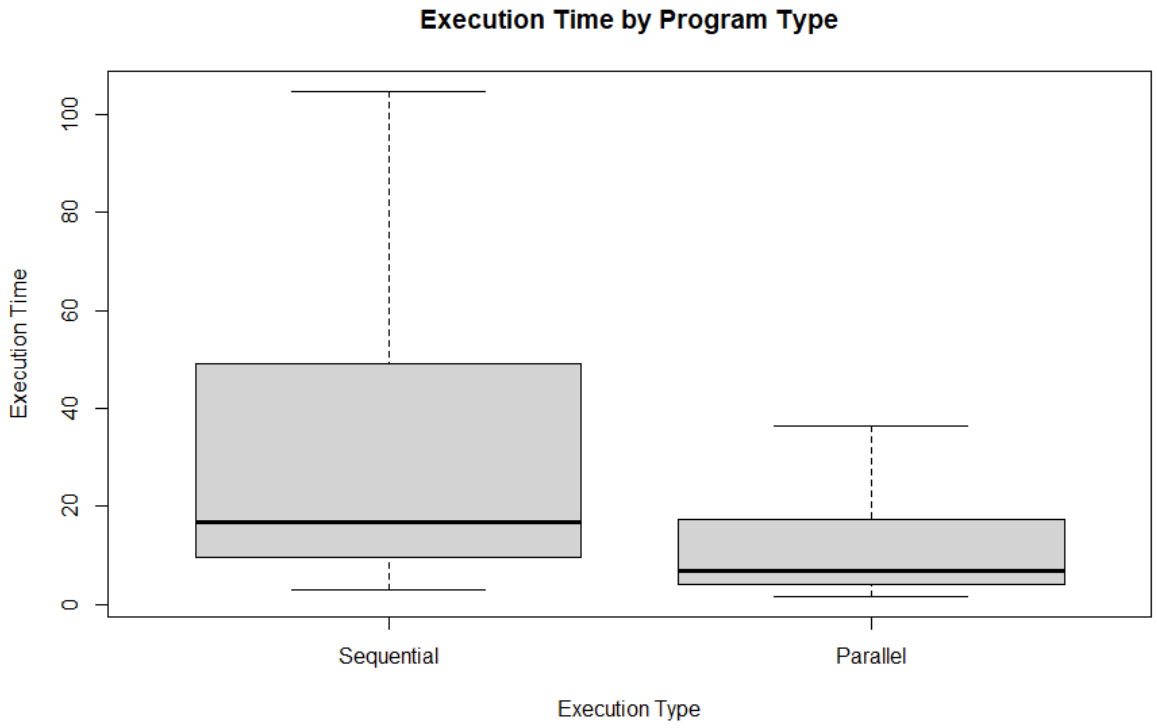

Fig 3.12, revised Boxplots after removal of outliers.