Analysing the time complexity and visual impact of the median cut algorithm for image quantization

Personal Code: kmj199

May 2024

Table of Contents

Introduction---------- 2
Rationale---------- 2
Background Information---------- 3
Aim---------- 5
Exploration---------- 6
Time Complexity---------- 6
SSIM---------- 11
Evaluation---------- 15
Strengths---------- 15
Weaknesses & Extensions---------- 16
Conclusion---------- 16
Appendix A---------- 17
1080p Image---------- 17
720p Image---------- 18
480p Image---------- 19

Introduction

Rationale

As a graphic designer, digital projects can grow in size quickly. With images growing to be thousands of pixels wide and tall, and displays growing to be even larger, the size of these files appear to grow exponentially as technology advances over the years. With this increase in resolution, however, images still should be able to be displayed on older, weaker technologies to ensure an equitable and accessible experience for all Internet users. Thus, image compression is employed to reduce the file size of an image, while maintaining the important details. An example of a JPEG-compressed image can be seen below in Figure 1.

Figure 1: Picture of flowers on left, and the same image but JPEG-compressed on right. File size is reduced.[1]

As my designs are affected by image compression through the online platforms I use, I decided to investigate the technologies behind it. To my surprise, there are a wide variety of algorithms used today, though one method of compression that I noticed was similar in many of these algorithms was *colour quantization*. Colour quantization is the process of reducing the number of colours in an image without losing quality[2]. Looking further into this method, I discovered the *median cut* (MC) *algorithm*, which, in the context of images, is a process that takes the red, green, and blue channel (RGB) data of an image and divides it into groups. Each group is assigned a specific colour from the image, thus quantizing the image and decreasing file size. This algorithm intrigued me, as it related to statistics and distributions from my SL Math curriculum, as the algorithm takes into account large datasets of numbers, and affects the various channels of the image (Red, Green, Blue), which can be observed through changes in distributions. In addition, studying this algorithm can give me deeper insight into its performance based on input size, or *time complexity*, which can be analysed using linear regression of its time complexity data. With the many mathematical concepts that combined to form a widely-used colour quantization algorithm, this investigation will serve as a great entry point into the wide field of graphics programming and image processing, enhancing the modern person's knowledge of how technology is always working around them.

---

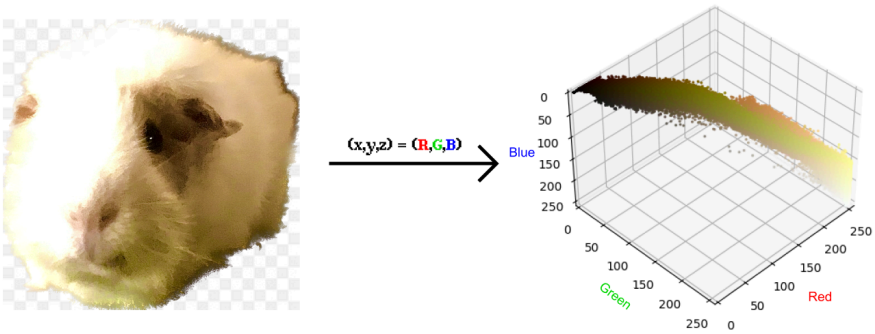
[1] Márquez-de-Silva, S., Felipe-Riverón, E., & Fernández, L. P. S. (2000). A Simple and Effective Method of Color Image Quantization. 750–759. https://link.springer.com/content/pdf/10.1007/978-3-540-85920-8_91.pdf

[2] Lazzerini, B., Pistolesi, F., & Lazzerini, B. (2000). Evaluation of the role of image resolution in the perception of color differences Evaluation of the role of image resolution in the perception of color differences. https://www.researchgate.net/figure/An-example-of-JPEG-compression-at-different-levels_fig2_341344498

Background Information

Prior to commencing this investigation, one must first understand the fundamental structure of MC. To improve clarity in the explanation of the algorithm, a 3D plot will be used, mapping each pixel's RGB values such that $(x, y, z) = (R, G, B)$. Figure 2 displays a sample image, and its respective 3D RGB plot. Note that $\{R, G, B \mid R, G, B \in Z,\ 0 \leq R, G, B \leq 255\}$. This is because the value of each channel is represented by an 8-bit binary value for each pixel when displaying the image. In other words, each channel has 2^8 or 256 possible values.

Figure 2: Sample input image on the left, and its 3D RGB plot on the right



On the first iteration of the algorithm, MC first finds which dimension on this 3D plot has the largest range. Table 1 below shows the ranges for each dimension, calculated by subtracting the minimum value from the max value of each channel. These values were found programmatically using a Python script.

Table 1: Ranges of RGB dimensions from Figure 2

	Red	**Blue**	**Green**
Minimum	0	0	0
Maximum	255	255	255
Range	255	255	255

Seen in Table 1, all three dimensions have an identical range. In this case, a random dimension is chosen, for instance, the green dimension. The median of the values in this dimension is taken, which is 128 rounded up in this instance. With this information, the dataset from Figure 1 is divided into 2 sub-datasets, one where the green value of all points is less than 128, and one where the green value is greater than 128. These plots are shown in Figure 3 below.

Figure 3: The two datasets generated by slicing the main one down the green median, displayed on an RGB plot

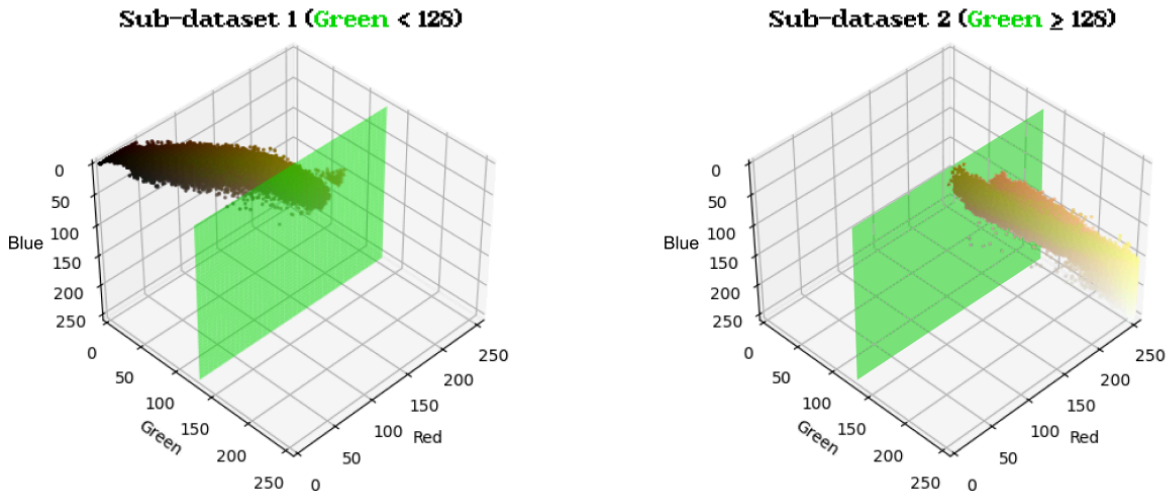


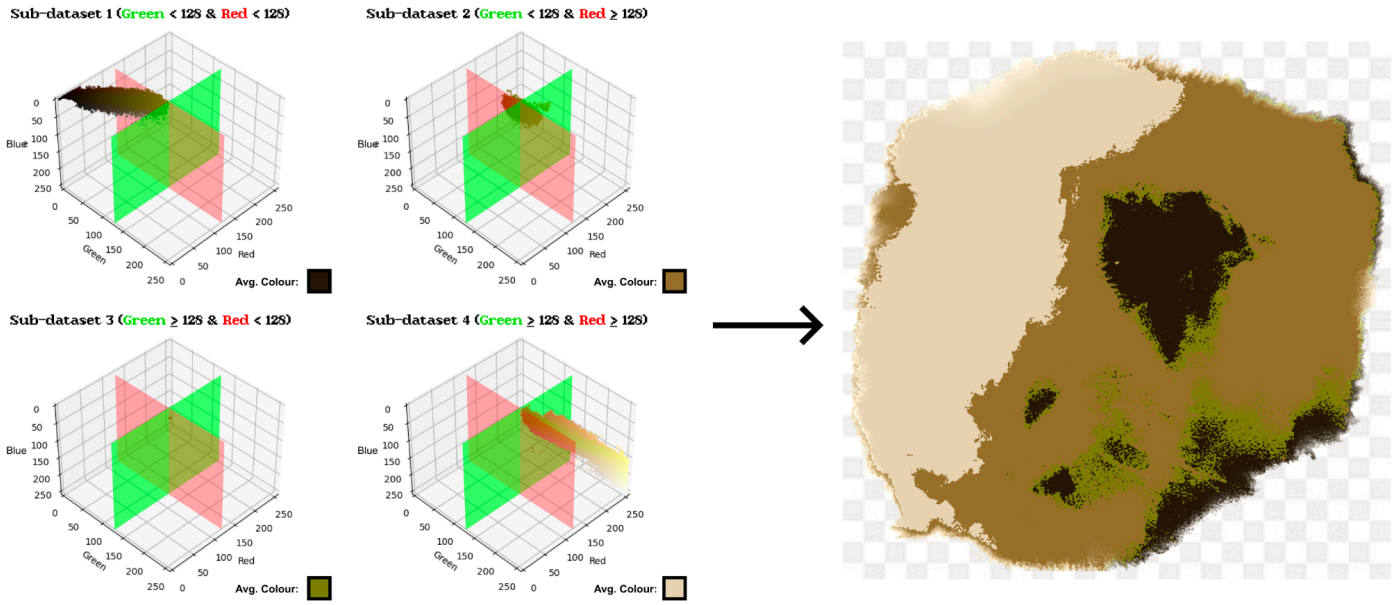
Table 2: Averages for each channel of each dataset

	Averages for channel			
	Red	**Green**	**Blue**	**Colour**
Dataset 1	34	24	8	
Dataset 2	145	113	53	
Dataset 3	125	128	31	
Dataset 4	228	212	182	

After this, the algorithm recursively repeats on each sub-dataset until the number of datasets equals the number of colours to quantize to. As each iteration cuts each sub-dataset into two parts, the size of the quantized palette will always be a power of two. After the algorithm completes, each dataset is assigned a specific colour, and all points are assigned their dataset's respective colour. Typically, the colour assigned to each dataset is the average colour of all the points in the dataset, calculated by averaging the red, green and blue channels separately and then combining each channel to make one colour.

In this particular example, a quantized palette of four colours is desired. Table 2 on the left shows the average values of each channel for each dataset and the subsequent colour generated, and Figure 4 below shows the plots for each dataset alongside the quantized image. Averages were calculated programmatically using a Python script.

Figure 4: All four subplots with their respective average colours, and the quantized image on the right



Aim

With the fundamental understanding of how the algorithm works, the investigation can begin. To best analyse the efficiency, visual impact, and error of the algorithm, two factors will be investigated. First, the *time complexity* (TC). TC is the measure of how the runtime of the algorithm scales based on input size[3]. This is often estimated by looking at the individual time cost of each step of the algorithm, and then aggregating this data to determine in what way it trends upwards. In the case of this investigation, the independent variable, x, will be the number of iterations of the algorithm to undergo, and the dependent variable, y, will be the runtime of the algorithm at a given x value. To improve the analysis, multiple trials will be conducted for each value of x, and the resolution of the image will be controlled to determine how the trend changes based on the size of the input file. The regression line of these plots will be calculated using least squares regression to determine the rate at which the time scales, thus giving a clear insight into the TC of the algorithm. The second factor to investigate will be the image's *Structural Similarity Index Measure* (SSIM), which is a method of calculating the visual similarity between two images, one uncompressed, and one compressed[4]. SSIM works by iterating over each pixel of the image, and calculating similarity data based on an $N \times N$ window centred at the specified pixel, returning a value in the interval $[0, 1]$. A value of 0 means there is no similarity, meaning the image has been altered drastically in that area, while a value of 1 means there is virtually no visual difference between the two images at that given area in the image. This measure takes into account the standard deviation, the variance, and the covariance of the values for either image, relying heavily on the field of statistics and distributions to determine the SSIM index.

[3] Olawanle, J. (2022, October 5). Big O Cheat Sheet – Time Complexity Chart. FreeCodeCamp.org. https://www.freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/

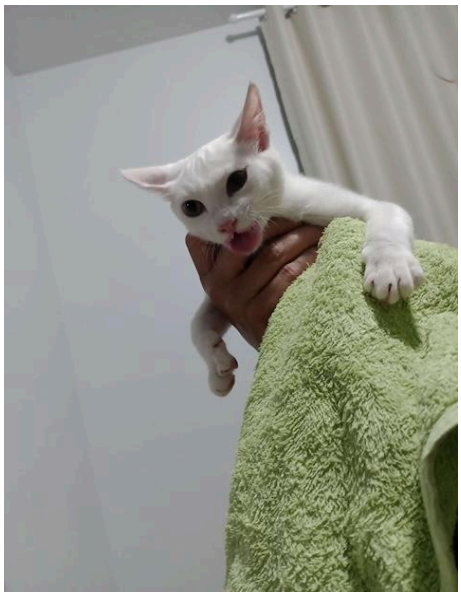
[4] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, 13(4), 600–612. https://doi.org/10.1109/tip.2003.819861

Exploration

Time Complexity

To begin, the image shown below in Figure 5 will be the image analysed during this exploration. This image was chosen for its varying level of detail around the image, as the blanket provides a high-detail, high contrast part of the image, while the wall is a very plain, low contrast part of the image, providing valuable insight into how MC affects the various complexities of a source image.

Figure 5: Image of a cat, which will be used for the MC quantization investigation



To obtain the best results, multiple trials of this algorithm were done for each datapoint, and three iterations of this experiment were done on three resolutions of this source image, 480×637 (480p), 720×956 (720p), and 1080×1434 (1080p) respectively, as to identify trends between both palette size and input file size on the TC of the algorithm. The full extent of the data can be found in Appendix A, but for ease of reference, calculations will be done with only the 1080p image's dataset with all figures rounded to 3 significant figures, which is shown below in Table 3. Values were recorded with all other applications on the device closed, as to ensure all system resources were dedicated towards this algorithm and to minimise external impact on the data.

Table 3: Table showing the runtime of the MC algorithm to complete on a 1080×1434 version of Figure 5

	Runtime (s)						
MC Iterations	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**
1	1.64	1.62	1.62	1.62	1.62	1.62	1.62
2	1.75	1.74	1.78	1.74	1.75	1.74	1.75
3	1.83	1.85	1.84	1.84	1.84	1.84	1.83
4	1.91	1.91	1.91	1.91	1.91	1.91	1.91
5	2.02	2.05	2.01	2.00	2.00	2.06	2.01
6	2.09	2.14	2.09	2.09	2.10	2.13	2.09
7	2.20	2.16	2.18	2.30	2.22	2.20	2.19
8	2.29	2.27	2.32	2.27	2.23	2.23	2.24

To begin, the average of all trials was taken for each datapoint to reduce the impact of random error on the results. This was done through the formula

$$\bar{x} = \frac{\sum_{i=1}^{7} x_i}{7}$$

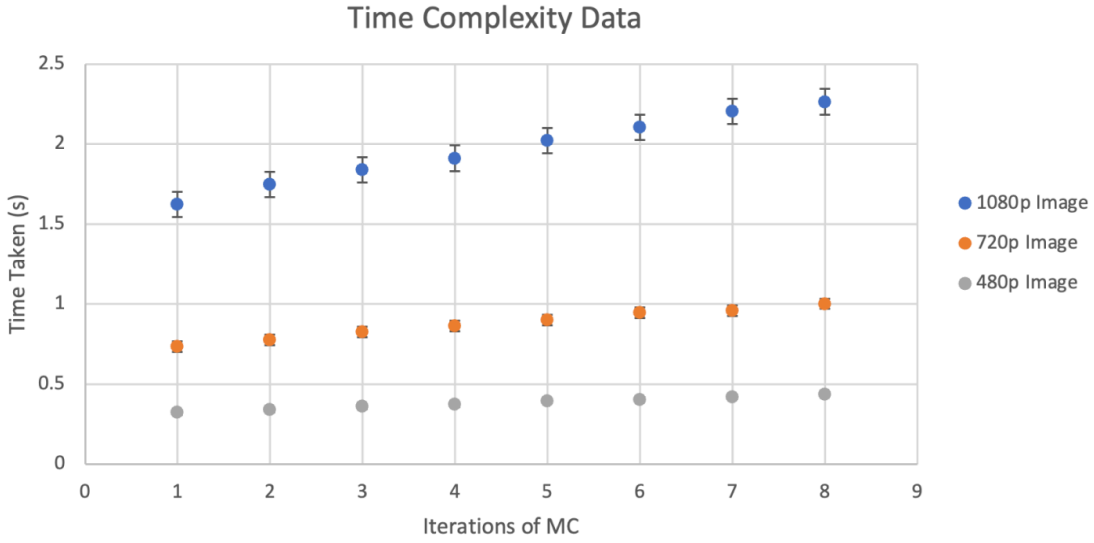
So, for 1 iteration of MC, the ave runtime would be:

$$\bar{x} = \frac{1.64+1.62+1.62+1.62+1.62+1.62+1.62}{7} = 1.62$$

Next, the data was charted on a scatter plot, with the iterations of MC as the independent variable, and the runtime in seconds as the dependent variable. The data for all resolutions is charted on the same plot, as to make trend comparison easier. Note that error bars were generated by making the lower bound the lowest trial value, and making the upper bound the highest trial value.

Figure 6 below shows the scatter plot generated with error bars.

Figure 6: Scatter plot of data from Appendix A with error bars



With the data appropriately recorded, the data must now be regressed to determine trends. Given the apparent linear trend observed in the above data, least squares regression is the most appropriate method to model this relationship. Least squares regression is a method of linear regression that minimises the sum of residual differences squared from the line of best fit, thus determining the most appropriate trend line for the data[5]. Sample calculations for the line of best fit will be done on the first 4 points of the 1080p image's data, as to improve clarity while still determining a relatively accurate trend line for the whole dataset. For these calculations, x will refer to the number of iterations of MC, and y will refer to the runtime based on the number of iterations. The formula to calculate the parameters of the line of best fit via least squares regression for the 1080p image's data is shown below:

$$m = \frac{4 \cdot [\sum_{i=1}^{4} (x_i y_i)] - \sum_{i=1}^{4} (x_i) \sum_{i=1}^{4} (y_i)}{4[\sum_{i=1}^{4} (x_i^2)] - [\sum_{i=1}^{4} (x_i)]^2}$$

$$b = \frac{\sum_{i=1}^{4} (y_i) - m \sum_{i=1}^{4} (x_i)}{4}$$

Therefore:

$$m = \frac{4 \cdot [(1 \cdot 1.62) + (2 \cdot 1.75) + (3 \cdot 1.84) + (4 \cdot 1.91)] - (1+2+3+4)(1.62+1.75+1.84+1.91)}{4 \cdot (1^2 + 2^2 + 3^2 + 4^2) - (1 + 2 + 3 + 4)^2}$$

$$m = \frac{4(18.28) - (10)(7.12)}{4(30) - 100}$$

$$m = \frac{73.1 - 71.2}{120 - 100}$$

$$m = \frac{1.9}{20.0}$$

$$m \approx 0.095$$

With the value of m, one can now find the value of the y-intercept, b:

[5] Spínola, D. (2020, September 8). The Least Squares Regression Method – How to Find the Line of Best Fit. FreeCodeCamp.org. https://www.freecodecamp.org/news/the-least-squares-regression-method-explained/

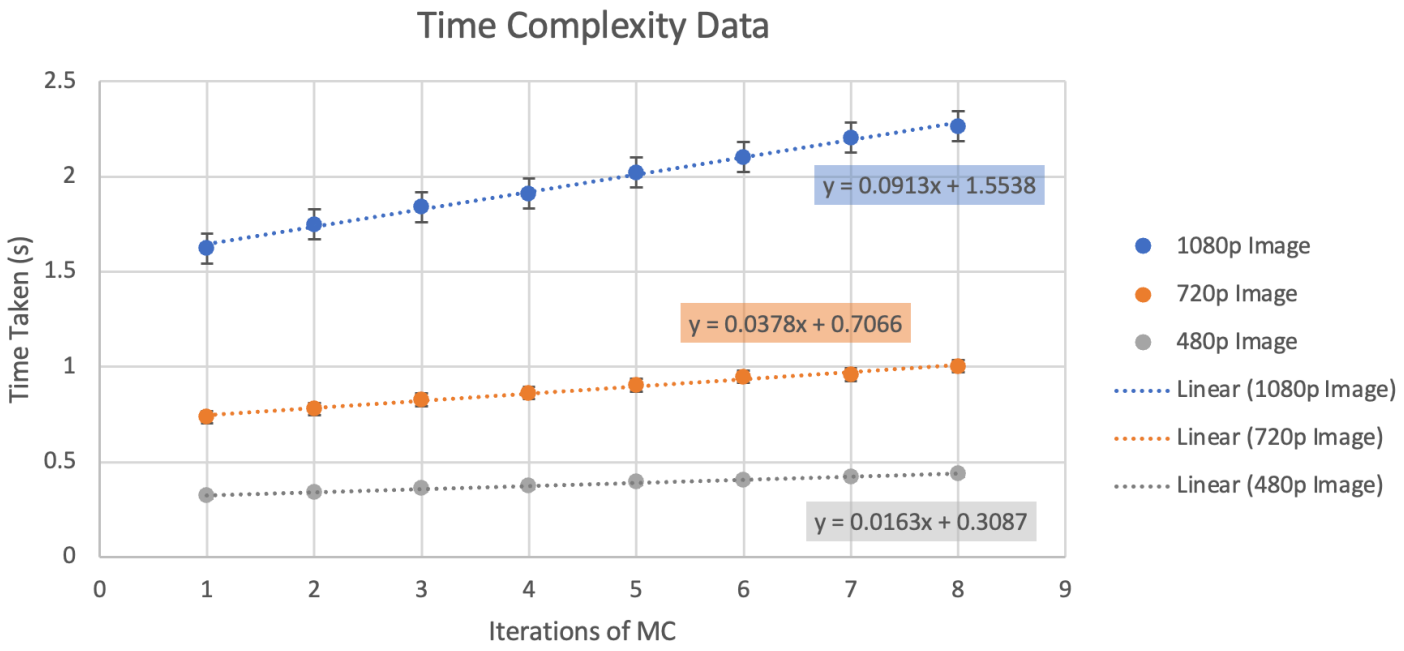
$$b \approx \frac{(1.62 + 1.75 + 1.84 + 1.91) - 0.095(1 + 2 + 3 + 4)}{4}$$
$$b \approx \frac{(1.62 + 1.75 + 1.84 + 1.91) - 0.095(1 + 2 + 3 + 4)}{4}$$
$$b \approx \frac{7.12 - 0.95}{4}$$
$$b \approx \frac{6.17}{4}$$
$$b \approx 1.54$$

Therefore, the equation of the regression line for the first 4 points of the 1080p image's data is:

$$y = 0.095x + 1.54$$

In Figure 7, the data from Appendix A is shown alongside each dataset's trend line. The equations for the lines are shown in the same colour as their respective trend line. Calculations for the equations of the trend line were done programmatically in Excel.

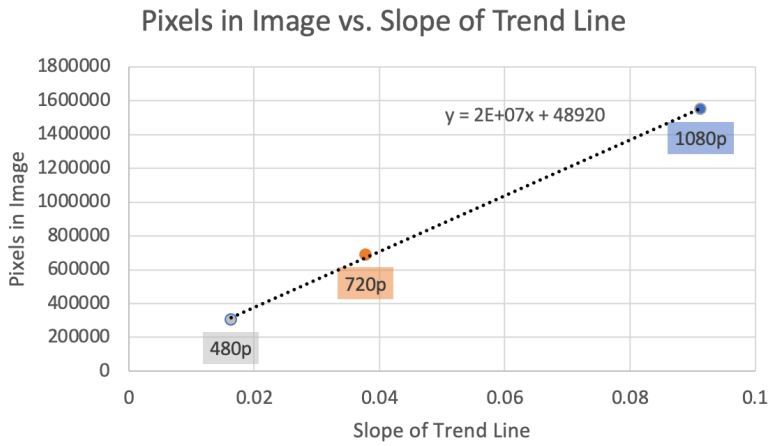
Figure 7: Scatter plot of data from Appendix A with trend lines and respective equations



In the plot above, one can see a general upwards trend among all of the image resolutions. In addition, the slope of the regression line increases as the resolution of the image does. To begin, one might notice that the trend lines indicate that the relationship between MC iterations and runtime is not directly proportional, implying that at an input size of 0, the algorithm will still take time to complete. Though it may seem peculiar, this is common among many algorithms, as there is often compilation time, which is the time it takes for the computer to translate programming languages into machine code, and initialization time, which is the time it takes to import necessary software packages, to take into account. Therefore, the y-intercept of each trend line is the initialization time of each run of the algorithm, changing depending on the resolution of the image as an increased resolution means an increase in time needed to load the image's data.

In addition, the slopes of the lines appear to follow a proportional relationship with the number of pixels in the specified image. A graph showing the slope of the trend line plotted against the number of pixels in the image is shown below in Figure 8, with the data's trend line and its respective equation.

Figure 8: Slope of the image's trend line plotted against the respective image's number of pixels



To determine the strength of this relationship, the data's coefficient of determination, denoted as r^2, will be calculated. To determine this value, the value of the Pearson correlation coefficient, denoted as r, is first needed. The formula to calculate r is as follows:

$$r = \frac{\sum_{i=1}^{3}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{3}[(x_i - \bar{x})^2] \cdot \sum_{i=1}^{3}[(y_i - \bar{y})^2]}}$$

x: *Slope of trend line*
y: *Number of pixels in image*

$$\bar{x} = \frac{0.0163 + 0.0378 + 0.0913}{3} = \frac{0.1454}{3} \approx 0.0485$$
$$\bar{y} = \frac{305760 + 688320 + 1548720}{3} = \frac{2542800}{3} = 847600$$

Therefore:

$$r = \frac{(0.0163 - 0.0485)(305760 - 847600) + (0.0378 - 0.0485)(688320 - 847600) + (0.0913 - 0.0485)(1548720 - 847600)}{\sqrt{[(0.0163 - 0.0485)^2 + (0.0378 - 0.0485)^2 + (0.0913 - 0.0485)^2] \cdot [((306760 - 847600)^2 + (688320 - 847600)^2 + (1548720 - 847600)^2}}$$

$$r = \frac{(-0.0322)(-541840) + (-0.0107)(-159280) + (0.0428)(701120)}{\sqrt{[(0.0322)^2 + (0.0107)^2 + (-0.0428)^2] \cdot [(-541840)^2) + (-159280)^2 + (701120)^2}}$$

$$r \approx \frac{17447.248 + 1704.296 + 30{,}007.936}{\sqrt{(1.03684\times10^{-3} + 1.1449\times10^{-4} + 1.83184\times10^{-3}) \cdot (2.935905856\times10^{11} + 2.53701184\times10^{10} + 4.915692544\times10^{11})}}$$

$$r \approx \frac{4.915948\times10^{4}}{\sqrt{(2.98317\times10^{-3})(8.105299584\times10^{11})}}$$

$$r \approx \frac{4.915948\times10^{4}}{4.9172641336\times10^{4}}$$

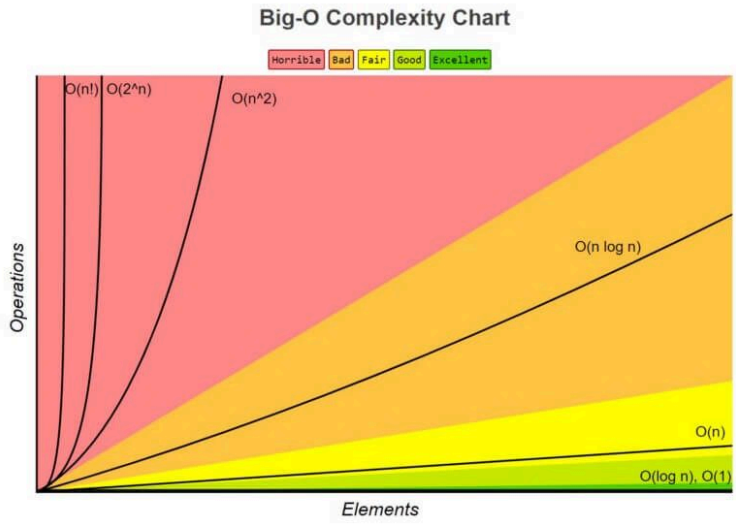
$$r \approx 1.000$$

Now, to determine the coefficient of determination, this value must be squared, giving a value of:

$$r^2 \approx 0.999$$

Evidenced through the coefficient of determination, there is a very strong linear relationship between the slope of each image's TC trend line and the number of pixels in the image. In addition, as the number of iterations in the algorithm increases, the runtime of the algorithm increases linearly. Thus, it can be concluded with minimal uncertainty that the TC of MC, represented in Big O notation, is $O(n)$. Big O notation is a form of mathematical notation that simplifies the relationship between input size, n, and runtime, $O(n)$, into a simple functional format[6]. Therefore, a relationship of $O(n)$ indicates that the algorithm's runtime scales linearly as the input size increases. Figure 9 below shows the relationship between various time complexities, rating them from Horrible to Excellent.

Figure 9: Graph of various time complexities denoted in Big O notation, rated from Horrible to Excellent [7]



As evidenced through the diagram, a TC of $O(n)$ is rated as a Good algorithm, and the only TC's better than $O(n)$ are $O(log\ n)$, found in algorithms like binary search, and $O(1)$, found in elementary operations like addition.

Compared to other image quantization methods and algorithms, a time complexity of $O(n)$ is highly desirable[8]. Therefore, one can conclude that MC, in terms of its time complexity, is a quite efficient manner of quantizing an image, making it a well-suited option for image processing and compression.

[6],[7] Huang, S. (2020, January 16). What is Big O Notation Explained: Space and Time Complexity. FreeCodeCamp.org. https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt-1674cfa8a23c/

[7] Huang, S. (2020, January 16). What is Big O Notation Explained: Space and Time Complexity. FreeCodeCamp.org. https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt-1674cfa8a23c/

[8] Gonzalez, R. C., Woods, R. E., & Masters, B. R. (2009). Digital Image Processing, Third Edition. Journal of Biomedical Optics, 14(2), 514–517. https://doi.org/10.1117/1.3115362

SSIM

Despite MC being a quite efficient algorithm, one must also assess its visual impact on the image to ascertain its suitability for digital images despite their complexity and resolution. To assess this, a measure known as the Structural Similarity Index Measure (SSIM) will be used. This function assesses the visual similarity between two images of matching resolution, A and B, by returning a value in the interval $[0, 1]$, 0 indicating minimal similarity, and 1 indicating high similarity[9]. The function iterates over every pixel of A, determining the SSIM of every pixel by analysing an $N \times N$ window surrounding the same pixel on both A and B. The function returns a 2D matrix with the same dimensions of the source images, each value in the matrix corresponding to the SSIM at its respective pixel. To determine the SSIM of an RGB image, the SSIM is calculated for each colour channel individually, and then averaged to determine the SSIM of A and B at a given pixel. SSIM is determined by three weighted constituent functions, shown below:

$$SSIM(A,B) = l(A,B)^{\alpha} \cdot c(A,B)^{\beta} \cdot s(A,B)^{\gamma}$$

Where:
$l(A,B)$: Luminance comparison measurement
$c(A,B)$: Contrast comparison measurement
$s(A,B)$: Structure comparison measurement
α: Luminance measurement weight
β: Contrast measurement weight
γ: Structure measurement weight

. The three functions that make up $SSIM(A,B)$ are defined as

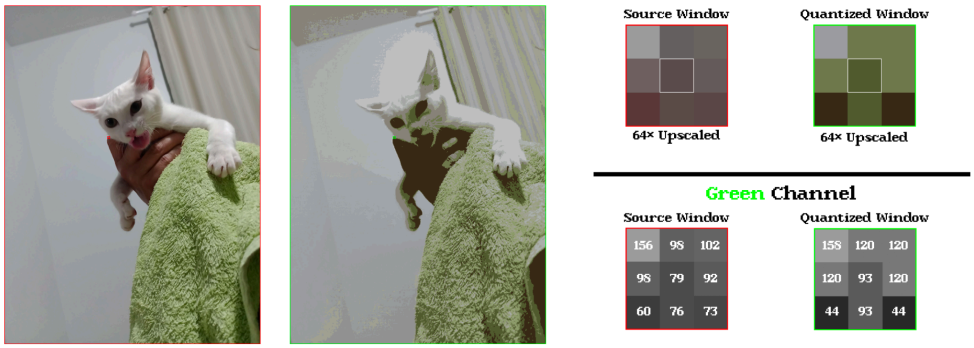
$$l(A,B) = \frac{2\mu_A\mu_B + c_1}{\mu_A^2 + \mu_B^2 + c_1}$$

$$c(A,B) = \frac{2\sigma_A\sigma_B + c_2}{\sigma_A^2 + \sigma_B^2 + c_2}$$

$$s(A,B) = \frac{\sigma_{AB} + c_3}{\sigma_A\sigma_B + c_3}$$

Where:
μ: Mean of window's data
σ^2: Variance
σ_{AB}: The covariance of A and B
$c_1: (k_1 L)^2$, $c_2: (k_2 L)^2$, $c_3 = \frac{1}{2}c_2$
$L = 255$, the channel's range
$k_1 = 0.01$, $k_2 = 0.03$

Figure 10: 480p source image next to its 4-iteration quantized version, with the windows upscaled



For clarity during sample calculations, a 3×3 window will be used and only the green channel will be taken into account. A randomly-chosen 3×3 window from both the source 480p image and its 4-iteration quantized counterpart is outlined and shown below in Figure 10. The pixel whose SSIM is to be calculated is outlined in white in the top right section. In the bottom right section, each pixel has been marked with its respective green channel value, visually represented by the brightness of the grey of each pixel.

[9] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, 13(4), 600–612. https://doi.org/10.1109/tip.2003.819861

With this information, the necessary values to determine the SSIM of this pixel can be calculated. To begin, the mean of the sample, μ, can be used to calculate the variance, σ^2, which is an average measurement of how far data is from the mean[10].

$$\sigma^2 = \frac{1}{9}\sum_{i=1}^{9}(p_i - \mu)^2$$

Where p_i is the green channel's value of the *i*-th pixel, arranged left-to-right, top-to-bottom.

Mean and Variance of Source Window

$$\mu_A = \frac{156 + 98 + 102 + 98 + 79 + 92 + 60 + 76 + 73}{9}$$
$$\mu_A \approx 92.7$$

$$\sigma_A^2 \approx \frac{(156 - 92.7)^2 + (98 - 92.7)^2 + (102 - 92.7)^2 + (98 - 92.7)^2 + (79 - 92.7)^2 + (92 - 92.7)^2 + (60 - 92.7)^2 + (76 - 92.7)^2 + (73 - 92.7)^2}{9}$$
$$\sigma_A^2 \approx \frac{4006.89 + 28.09 + 86.49 + 28.09 + 187.69 + 0.49 + 1069.29 + 278.89 + 388.09}{9}$$
$$\sigma_A^2 \approx \frac{6074.01}{9}$$
$$\sigma_A^2 \approx 674.89$$

Mean and Variance of Quantized Window

$$\mu_A = \frac{158 + 120 + 120 + 120 + 93 + 120 + 44 + 93 + 44}{9}$$
$$\mu_A \approx 101.3$$

$$\sigma_B^2 \approx \frac{(158 - 101.3)^2 + (120 - 101.3)^2 + (120 - 101.3)^2 + (120 - 101.3)^2 + (93 - 101.3)^2 + (120 - 101.3)^2 + (44 - 101.3)^2 + (93 - 101.3)^2 + (44 - 101.3)^2}{9}$$
$$\sigma_B^2 \approx \frac{3214.89 + 349.69 + 349.69 + 349.69 + 68.89 + 349.69 + 3283.29 + 68.89 + 3283.29}{9}$$
$$\sigma_B^2 \approx \frac{11318.01}{9}$$
$$\sigma_B^2 \approx 1257.56$$

[10] Statistics Canada. (2021, September 2). Statistics: Power from Data! Variance and standard deviation. Statcan.gc.ca. https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch12/5214891-eng.htm

Next, the covariance, σ_{AB}, which measures how two lists of paired values rise or decrease together [11], must be found using the formula:

$$\sigma_{AB} = \frac{1}{9}\sum_{i=1}^{9}(p_{A_i} - \bar{p_A})(p_{B_i} - \bar{p_B})$$

$$\sigma_{AB} \approx \frac{1}{9}[(156 - 92.7)(158 - 101.3) + (98 - 92.7)(120 - 101.3) + (102 - 92.7)(120 - 101.3) + (98 - 92.7)(120 - 101.3) + (79 - 92.7)(93 - 101.3) + (92 - 92.7)(120 - 101.3) + (60 - 92.7)(44 - 101.3) + (76 - 92.7)(93 - 101.3) + (73 - 92.7)(44 - 101.3)]$$

$$\sigma_{AB} \approx \frac{1}{9}(3589.11 + 99.11 + 173.91 + 99.11 + 113.71 - 13.09 + 1873.71 + 138.61 + 1128.81)$$

$$\sigma_{AB} \approx \frac{1}{9}(7202.99)$$

$$\sigma_{AB} \approx 800.3$$

With the above information, the value of SSIM's constituent functions can be calculated.

Luminance Comparison

$$l(A,B) = \frac{2\mu_A\mu_B + c_1}{\mu_A^2 + \mu_B^2 + c_1}$$

$$l(A,B) \approx \frac{2(92.7)(101.3) + (0.01 \cdot 255)}{(92.7)^2 + (101.3)^2 + (0.01 \cdot 255)}$$

$$l(A,B) \approx \frac{18781.02 + 2.55}{18854.98 + 2.55}$$

$$l(A,B) \approx 0.9960779593$$

Contrast Comparison

$$c(A,B) = \frac{2\sigma_A\sigma_B + c_2}{\sigma_A^2 + \sigma_B^2 + c_2}$$

$$c(A,B) \approx \frac{2(\sqrt{674.89})(\sqrt{1257.56}) + (0.03 \cdot 255)}{(674.89) + (1257.56) + (0.03 \cdot 255)}$$

$$c(A,B) \approx \frac{1842.51 + 7.65}{1932.45 + 7.65}$$

$$c(A,B) \approx 0.9536415649$$

Structure Comparison

$$s(A,B) = \frac{\sigma_{AB} + c_3}{\sigma_A\sigma_B + c_3}$$

$$s(A,B) \approx \frac{(800.3) + \frac{1}{2}(7.65)}{(\sqrt{674.89})(\sqrt{1257.56}) + \frac{1}{2}(7.65)}$$

$$s(A,B) \approx \frac{800.3 + 3.825}{921.25 + 3.825}$$

$$s(A,B) \approx 0.8692471605$$

Finally, the SSIM of the specified pixel can be calculated. As for the weights of each component function, α, β, and γ, they will all be valued at 1, the default value, as there is no need to weight any one of these comparison measurements more than the other.

$$SSIM(A,B) = l(A,B)^1 \cdot c(A,B)^1 \cdot s(A,B)^1$$

$$SSIM(A,B) \approx 0.9960779593 \cdot 0.9536415649 \cdot 0.8692471605$$

$$SSIM(A,B) \approx 0.83$$

[11] Weisstein, E. W. (2024, February 21). Covariance. Mathworld.wolfram.com. https://mathworld.wolfram.com/Covariance.html

In Figure 11 below, the SSIM of all the compressed images generated from the 1080p image, from 1 iteration of MC to 8 iterations of MC, is calculated. Each pixel's SSIM value is visually represented through its intensity on a heat map, blue indicating a high structural similarity, and red indicating a low structural similarity. The window is sized at 11×11 pixels, as to get better insight on wider structural differences in the image. Table 4 below shows the average SSIM for each image resolution, which can be found in Appendix A. 3 significant figures are shown to improve clarity in mean SSIM comparison. The data for only the 1080p image is shown, as there are highly similar structural differences and mean SSIM values between all resolutions, making it redundant to show all of the data.

Figure 11: SSIM heat map indicating the structural similarity between the source image and varying iterations

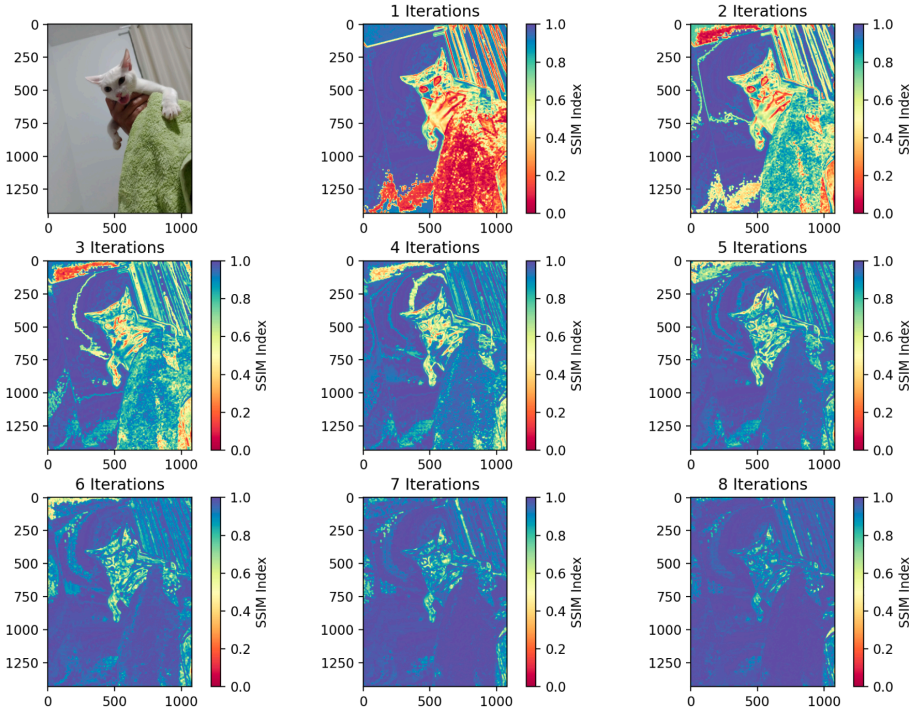


Table 4: The mean SSIM for 1080p images ranging from 1 to 8 iterations of MC (see Appendix A)

Number of Iterations	Mean SSIM Index
1	0.630
2	0.767
3	0.840
4	0.894
5	0.915
6	0.932
7	0.954
8	0.967

As can be seen, MC suffers in high-contrast, high-detail areas like the towel and the cat's face at lower iterations, returning a SSIM value roughly in the interval of $[0, 0.4]$. At these lower iterations, however, flatter, low-detail areas like the wall are highly similar, with values in the interval of $[0.8, 1.0]$. As the number of iterations increase, however, the heat maps noticeably get less red, with values in the 6-iteration heatmap in the interval $[0.6, 1.0]$. Given the algorithm's quick runtime at 1080p, and a mean SSIM greater than 0.9 at only 5 iterations, the data above demonstrates the high efficiency and quality of this colour quantization algorithm, as it maintains a high degree of structural similarity in a short amount of time and number of iterations.

Evaluation

As shown through the above data, MC is a highly efficient and effective algorithm for image quantization, as its $O(n)$ time complexity, its short runtime, and its high SSIM value at a low number of iterations demonstrate high levels of structural similarity in only a few iterations of the algorithm. Given the 1080p image quantization's short runtime, it is clear that as image processing technology continues to develop, MC will remain to be a very viable option for colour quantization in the image compression process.

Throughout the process, calculations were computed programmatically, either with a Python script or with Excel formulae, thus cementing my results as reliable throughout every step of the exploration process. However, with calculations taking into account only a small sample of the total dataset, some numbers may not be fully reflective of the data as a whole. However, these discrepancies are minimal, such as the slope in the least squares regression calculation, and similar conclusions can be reached even while using a small subset of the original data. To best reflect on this investigation, its strengths, weaknesses, and extensions will be listed below in a tabular format.

Strengths

Strength	Significance
Number of trials	By undergoing 7 trials for each iteration of the algorithm for each resolution, the impact of random error was minimised drastically, providing much more accurate and precise results by taking the average of this data
Programmatic data collection	The modern computer is, as its name suggests, a highly precise machine to compute calculations. By collecting data with Python scripts, highly precise figures up to 15 decimal places were collected, providing me with highly precise calculations in return.
Controlling external factors	To ensure results were not affected by external factors, many measures were taken into account. Foremost, the images loaded into the algorithm were all PNG files in a BGR colour space. A colour space is a specific organisation of colours that images can be represented in, which can affect how the image is read by a program [12]. By encoding all the images in BGR, and by maintaining a PNG file format, there was no impact on the collected data from how information is stored in various file formats and how the colour data was stored.

[12] Kelly, C. (2020, February 3). The Essential Guide to Color Spaces. Frame.io Insider. https://blog.frame.io/2020/02/03/color-spaces-101/

Weaknesses & Extensions

Weakness	Significance	Extensions
Limited diversity in exploration	Though an image with varying amounts of detail and contrast was used, the data collected solely on this image may not be reflective of how MC impacts all images.	In future explorations, a variety of images can be used, and individualised conclusions can be drawn depending on the type of image MC is being used on
Lack of human-perception based data	Though SSIM was used to determine the structural similarity between a compressed and an uncompressed image, qualitative data from the people who will be viewing these images would be highly valuable in determining the viability of MC as images continue to grow in size.	In addition to the mathematical data collected, qualitative data collected from willing participants can be used to draw more informed conclusions on the applicability of MC in the future.
Lack of comparison to other quantization algorithms	For the intentions of this report, only MC was investigated. However, to provide a more informed conclusion on the future relevance of the algorithm, the performance of this algorithm could be compared to others like octree clustering and population.	To add to the data of this study, other quantization algorithms can be studied, allowing for a more comparative analysis between them and providing more insight into the value of MC.

Conclusion

Overall, this exploration was very insightful and interesting, as it provided me a deeper look into the inner functions of the technology that most people don't even notice in their daily use of the Internet. Not only did I understand the inner workings of MC, but I understood the interesting mathematical formulae used in both its execution and subsequent analysis, such as through SSIM. Undergoing this investigation has brought to light the importance of statistics and distributions in my daily life, pushing me to explore other image compression algorithms and their respective functions and weaknesses.

Appendix A

1080p Image									
	Time Taken								
MC Iterations	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Mean SSIM Index	Average Time Taken
1	1.641304016	1.616406918	1.620800018	1.618023157	1.6240201	1.622184992	1.617402077	0.669209467	1.622877325
2	1.747537136	1.736216784	1.784763098	1.739622116	1.74783802	1.739616871	1.745158911	0.779706158	1.748678991
3	1.833168983	1.845149755	1.844254017	1.837546825	1.842844963	1.84493804	1.83484292	0.84463235	1.840392215
4	1.91187191	1.910714149	1.912487745	1.912235022	1.9099648	1.913146019	1.908067942	0.893808808	1.911212512
5	2.022443056	2.049596786	2.00521183	2.003340006	1.999572992	2.060355663	2.012772083	0.915286779	2.021898917
6	2.087996006	2.136901855	2.087761879	2.088424921	2.101873875	2.128153801	2.090166092	0.929516383	2.103039776
7	2.19707489	2.155225754	2.179836988	2.298461914	2.219496012	2.195509911	2.189277887	0.951522054	2.204983337
8	2.286597252	2.269472122	2.319741011	2.272506237	2.234490156	2.226951122	2.238789082	0.964005854	2.26407814

720p Image									
	Time Taken								
MC Iterations	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**	**Mean SSIM Index**	**Average Time Taken**
1	0.737879038	0.740190983	0.737089157	0.733402729	0.734052896	0.735846996	0.731946945	0.640967885	0.735772678
2	0.774497986	0.776885986	0.775827885	0.779205084	0.772104979	0.789646864	0.777001858	0.770635271	0.77788152
3	0.824280977	0.82906127	0.846982241	0.824938774	0.817530155	0.82799077	0.817332029	0.844044418	0.826873745
4	0.85915184	0.863956928	0.863765955	0.872402906	0.861654758	0.857630968	0.85841012	0.895127201	0.862424782
5	0.907071114	0.919595003	0.895959854	0.902408838	0.90191102	0.899218798	0.891669035	0.914069061	0.902547666
6	0.925637007	0.933531284	1.027480841	0.930001974	0.944261074	0.936269045	0.938866854	0.933173509	0.948006868
7	0.955240011	0.955998898	0.961159945	0.957256079	0.958635092	0.969009161	0.959115982	0.95519113	0.959487881
8	0.994009018	1.009194136	1.000723124	1.006062269	0.995697737	1.021483898	0.990911007	0.968078646	1.002583027

480p Image									
	Time Taken								
MC Iterations	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Trial 6**	**Trial 7**	**Mean SSIM Index**	**Average Time Taken**
1	0.322583914	0.324057102	0.329357147	0.323922873	0.316299915	0.324368238	0.317339897	0.606134084	0.322561298
2	0.341840982	0.341628313	0.338770151	0.340436935	0.340285778	0.338228941	0.341022015	0.756326293	0.340316159
3	0.359313965	0.377938986	0.357146978	0.359540939	0.356750727	0.356410742	0.356603146	0.838225856	0.360529355
4	0.372189045	0.374481201	0.371733904	0.370991945	0.374619961	0.373448133	0.374111891	0.893208296	0.373082297
5	0.390592098	0.390578985	0.38990593	0.417810917	0.394851208	0.391924143	0.389471292	0.915266051	0.395019225
6	0.40332818	0.403142929	0.403081179	0.404591084	0.406962872	0.409881115	0.40472579	0.931945169	0.405101878
7	0.421208858	0.421382904	0.420855999	0.418799877	0.418676853	0.426696777	0.423764944	0.962766954	0.421626602
8	0.436728001	0.438790083	0.438549995	0.436202288	0.439990044	0.437893152	0.435921192	0.975038251	0.437724965