Indian Institute of Technology Indore

Dominant Color Palette Extraction by Clustering Algorithms and Reconstruction of Image

Computational Intelligence (CS354N)

Author
Aniket Sangwan (180001005)
Sarthak Jain (180001047)
Sundesh Gupta (180001057)

May 14, 2021

Contents

Chapter 1

Introduction

1.1 Problem Definition

Color Quantization problem consists of two steps:

- **Color Palette Generation:** A reduced number of palette colors are specified which are later used to represent the original image.
- **Pixel Mapping:** Each pixel in the original image is mapped to its nearest palette color and quantized image is generated.

1.2 Design and Analysis of Problem

Our main goal is to reduce the storage requirement and speedup the transfer time of images. The bandwidth limits the applications of multimedia in today's world. Color quantization is important when we need to transfer a set of large images over the internet. It ramps up the efficiency and processing speed by solving storage and transmission related problems.

Color quantization algorithms can be primarily grouped into two categories: splitting algorithms and clustering based algorithms. Splitting algorithms split the color space of the original image into two separate subspaces according to some preference criteria. Splitting is performed until the expected number of subspaces are obtained. Clustering algorithms perform clustering of the color space, where each cluster's representative is chosen as a palette color. The cluster representatives are iteratively updated using specific algorithms.

In general, splitting algorithms are fast. The disadvantage is that generally no global optima are obtained, because a decision made for splitting at one level cannot be undone at a further level. Clustering algorithms, on the other hand, provide better result but are more time consuming. Also, their dependence on the initial conditions makes it difficult to get the desired solution.

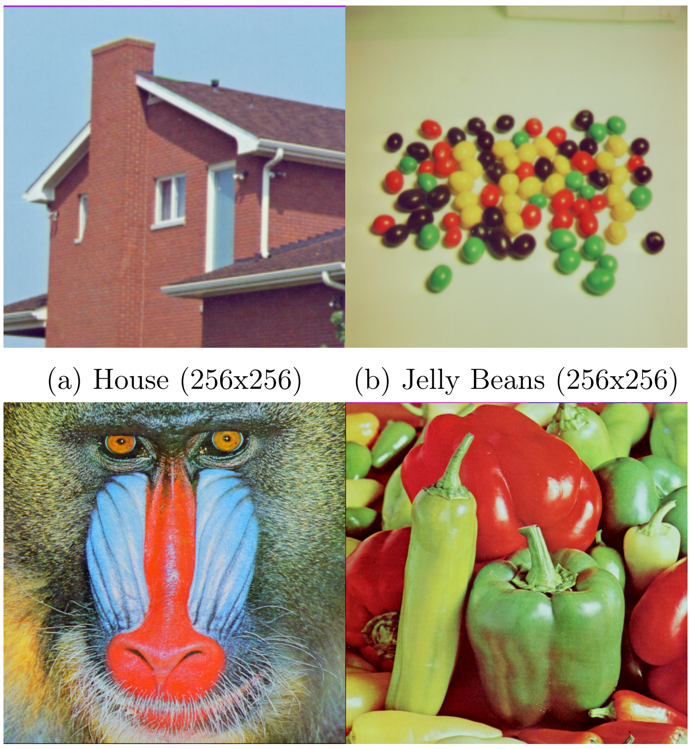
We have implemented three algorithms: K-means clustering algorithm, Adaptive clustering algorithm and Hierarchical Competitive learning and evaluated them using an image set.

Chapter 2

Data Collection and Preprocessing

2.1 Data Collection

USC-SIPI Image Database is a popular collection of digitized images used in research in image processing, image analysis, and computer vision. Figure 2.1 shows the 4 images that are used to test the performance of the algorithms used.



(a) House (256x256) (b) Jelly Beans (256x256)

(c) Baboon (512x512) (d) Peppers (512x512)

Figure 2.1: Images from the USC-SIPI Dataset

2.2 Preprocessing

Preprocessing of the given dataset includes the formation of 3D RGB Histogram. This is a table generated to pick the most dominant color for the given image and use it as the initial cluster. The process includes formation of bins of the most dominant colors. The histogram is generated as follows :

1. N_b bits of each color component for a given pixel are set to zero which reduces the execution time and also helps in easily locating the dominant colors in the images.
2. All unique colors are extracted from the image after performing the first step and the number of pixels for each color is stored.
3. The unique color array is sorted in descending order of the frequency of each color.
4. Each color is then picked and if the frequency of the color is greater than the value of N_p, the color is selected as one of the bin.
5. The generated bins are finally sorted in the descending order of frequency.

The array of sorted bins that is generated by following the above steps is called the 3D RGB Histogram.

Chapter 3

Study and Understanding of Algorithms

3.1 K-means algorithm

It is a clustering algorithm which tries to minimise the distance of the points in a cluster with their representative (cluster center). The steps involved are:

1. Select 'k' cluster centers randomly.
2. Calculate the distance between each point and each cluster center.
3. Assign the point to its closest cluster center (E-step).
4. Assign new value to each cluster center by calculating the mean of all the points belonging to the cluster (M-step).
5. Recalculate the distance between each point and newly obtained cluster centers.
6. If any point is reassigned then go to step 3, else stop the algorithm.

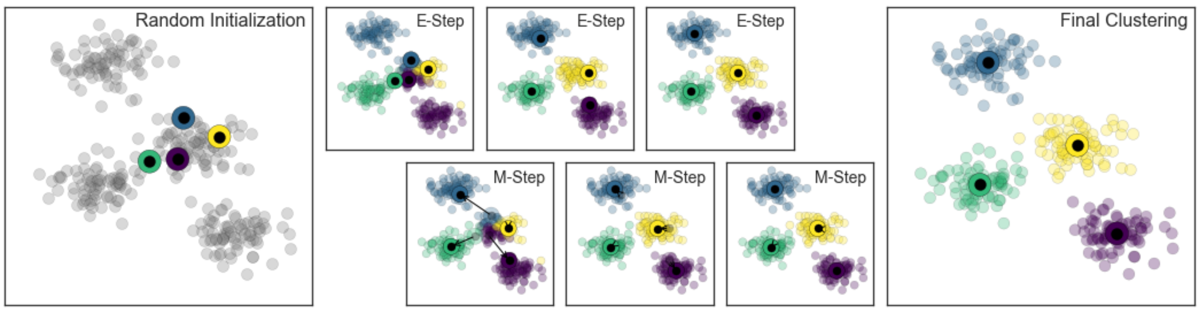


Figure 3.1: K-Means clustering

Under typical circumstances, each iteration will always result in a better estimate of the cluster characteristics.

3.2 Hierarchical Competitive learning (HCL)

HCL is a clustering algorithm which uses the Competitive Learning (CL) algorithm, along with the classical splitting algorithm like Median-Cut algorithm. Traditional splitting algorithms have a disadvantage of being stuck at a local minima. Since splitting algorithms do not need initial conditions, a way to overcome the problem of local optima is by combining CL and splitting techniques. The steps involved in HCL are:

1. Start with one cluster representative at a random position and apply CL forming 1 cluster
2. Degenerate or split this center into two independent ones, and apply CL again to both centers.
3. Repeat step 2 above for all obtained cluster centers, until the desired number of clusters is obtained.

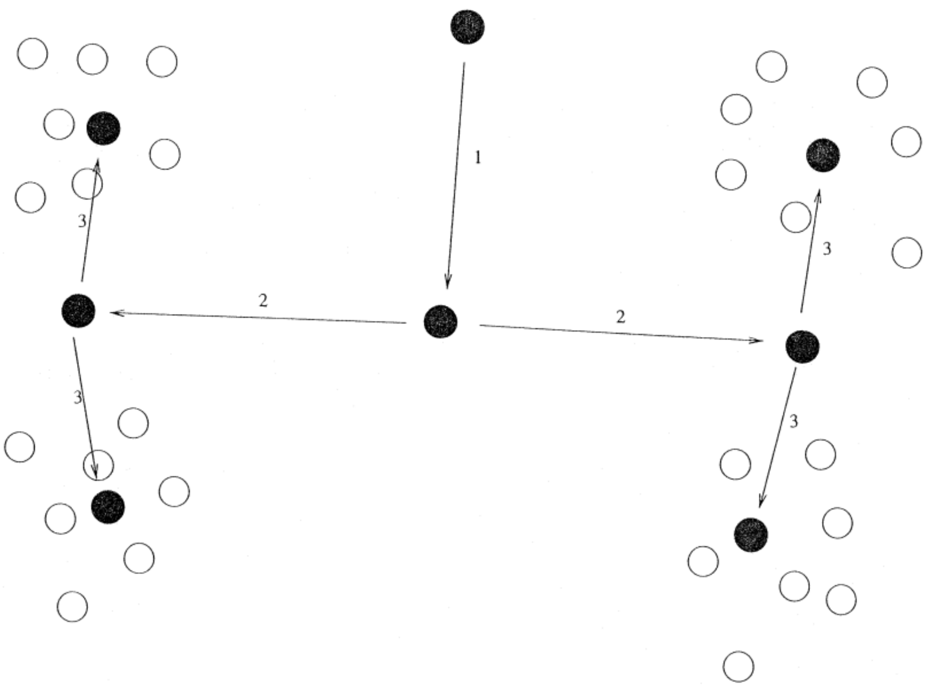


Figure 3.2: The three iteration steps in Hierarchical Clustering Algorithm

3.3 Adaptive Clustering Algorithm

Adaptive Clustering Algorithm is a clustering algorithm that involves the rationale of Maximin clustering Algorithm. This algorithm operates on the 3D RGB Histogram generated during preprocessing. The algorithm can overcome the time consuming problem of the general clustering algorithm. Also, this algorithm does not need any initial guess for a cluster, thus the condition where the algorithm gets stuck on a local minima is not reached. The sorted histogram bin list generated in the preprocessing phase helps in the reducing the runtime since the bins are already sorted and less in number than the total number of unique pixels.

The colors in the final color palette are called classes in the algorithm. The algorithm works as follows :

1. The first bin in the histogram is set as the first class (dominant color).
2. Select the bin with largest frequency other than the selected ones as the next bin. Calculate the color distance between the selected bin and its nearest dominant bin. If the color distance is lesser than the threshold η, the selected color is dropped and we move to the next most frequent color and repeat the same step. If the distance is greater than the threshold, the selected color is set as the next class (dominant color).
3. The above step is repeated until we reach the required number of classes (K). If all the bins are traversed and the required number of classes is not reached, the value of η is decreased and the algorithm restarts.
4. All the unassigned bins are then assigned the nearest class and serve as the neighbours for the class.
5. The final value of R, G and B components for a class is calculated as the weighted average of all the members of the class (the frequency of each color in a class serves as the weight).
6. The colors of the original image are mapped to the nearest palette color. The naive algorithm is time consuming thus the mapping is done using bitmasking to reduce the runtime of mapping process.

The final palette colors are obtained after step 5.

The following notations and methods are used in the above mentioned algorithm :

- K is the number of colors we wish to quantize in the original image.
- The initial value of η is set to $\sqrt[3]{\frac{255^3}{K}}$
- If the number of unfound classes in step 3 is greater than 64, η is decreased 4. Otherwise η is decreased by 2.
- The color distance between color i and color j is calculated as

  codist(i, j) $= \sqrt{(R_i - R_j)^2 + (G_i - G_j)^2 + (B_i - B_j)^2}$

Chapter 4

Performance Measurement Criteria

1. **Quantization Error:** Quantization error is a metric of the error introduced by moving each point from its original position to its associated cluster point. We are measuring this error as the sum of the square of the euclidean distance of each point from its cluster center. It is given by:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}[(R_i - \widehat{R}_i)^2 + (G_i - \widehat{G}_i)^2 + (B_i - \widehat{B}_i)^2]$$

2. **CPU time:** The average CPU time for each image in the dataset is calculated on a fixed workstation to compare the convergence time of each algorithm.
3. **Dependence on initial conditions:** Distribution of Quantization error is calculated after 100 independent runs using random initial conditions.

Chapter 5

Experimentation and Results

We have taken 4 images, namely House, Jellybeans, Baboon and Peppers from UCI-SIPI dataset. Out of these, House and Jellybeans have height and width of 256x256 pixels, whereas Baboon and Peppers have height and width of 512x512 pixels. Also, each image is tested over 4 different counts of quantised colors i.e. 16, 32, 64 and 128.

5.0.1 Testing Run-Times

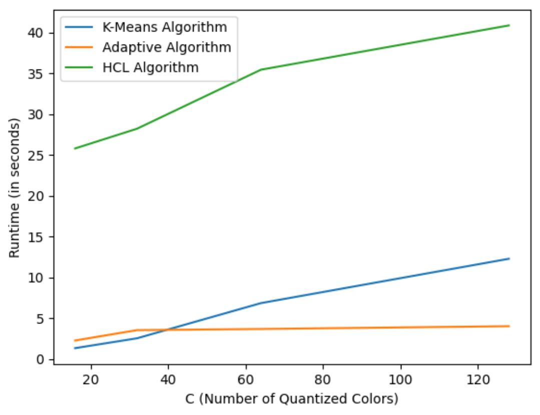
Table 5.1 and Figure 5.1 shows the comparison of run-times of each algorithm, over different images and different number of quantised colors.

	Runtime (in sec)			
Image	C	K-Means	Adaptive	HCL
House	16	1.33	2.27	25.80
	32	2.54	3.54	28.22
	64	6.85	3.68	35.45
	128	12.28	4.02	40.87
Jellybeans	16	1.29	2.36	19.72
	32	2.43	3.31	26.78
	64	5.47	3.64	28.23
	128	9.86	4.20	33.10
Baboon	16	5.12	8.55	90.50
	32	12.48	14.23	111.65
	64	21.29	15.72	128.98
	128	42.37	23.99	148.71
Peppers	16	7.56	6.40	91.16
	32	11.54	7.72	108.86
	64	20.79	10.32	125.25
	128	37.00	12.11	152.52

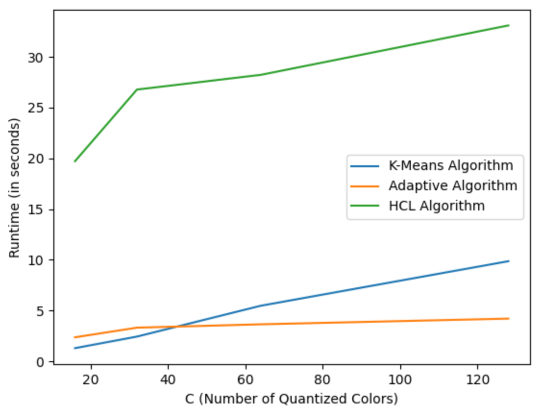
Table 5.1: Obtained runtime (in sec) for each image in the dataset

5.0.2 Testing Mean Squared Error

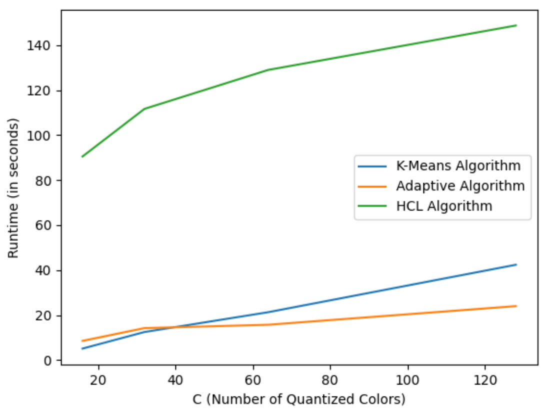
Table 5.2 and Figure 5.2 shows the comparison of MSE (Quantization Error) of each algorithm, over different images and different number of quantised colors.



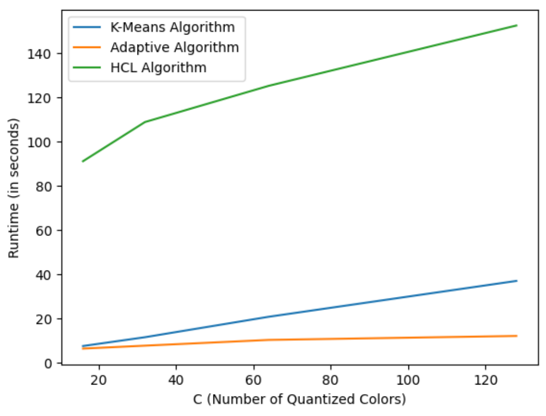
(a) House



(b) Jellybeans



(c) Baboon



(d) Peppers

Figure 5.1: Runtime Plots

5.0.3 Testing Dependence on Initial Conditions

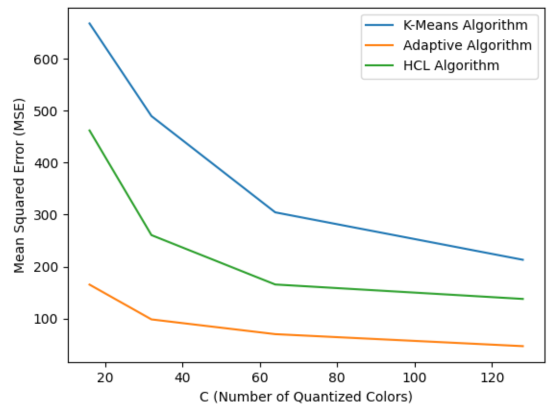
To test the dependence of K-Means, HCL and Adaptive clustering algorithm on initial conditions, we ran each algorithm 100 times for the House image setting the number of quantized colors to 16. Figure 5.3 shows the change in MSE over different iterations.

5.0.4 Visualising Quantized Images

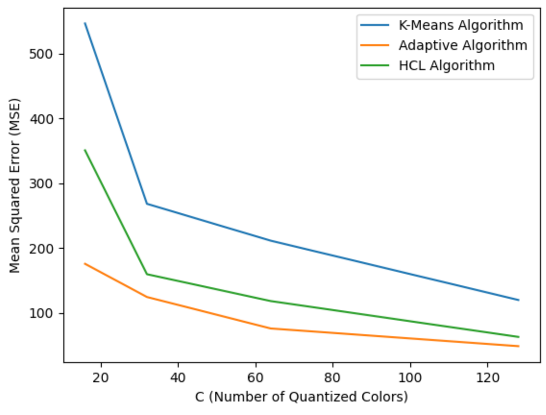
Figure 5.4, Figure 5.5 and Figure 5.6 shows the quantised images of K-means, HCL and Adaptive Clustering algorithms respectively.

		MSE		
Image	C	K-Means	Adaptive	HCL
House	16	667.99	165.35	462.06
	32	489.71	98.39	260.63
	64	304.43	69.99	165.56
	128	213.12	46.89	137.80
Jellybeans	16	546.33	175.57	350.63
	32	268.15	124.39	159.55
	64	211.26	75.83	118.12
	128	119.79	48.56	62.83
Baboon	16	1179.61	776.36	1048.37
	32	974.50	462.82	705.79
	64	658.60	294.91	413.53
	128	395.79	180.67	291.85
Peppers	16	1078.71	552.11	838.51
	32	766.20	322.54	528.78
	64	390.07	183.65	299.63
	128	253.02	121.39	188.88

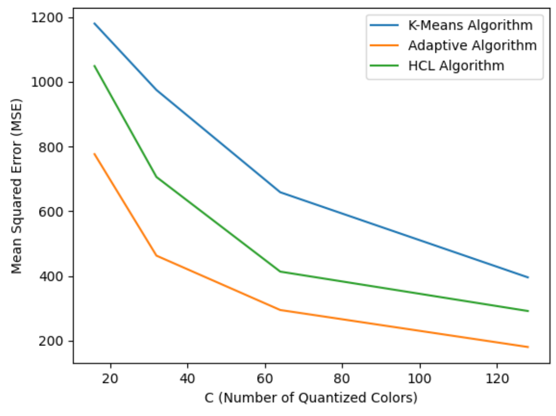
Table 5.2: Obtained MSEs for each image in the dataset



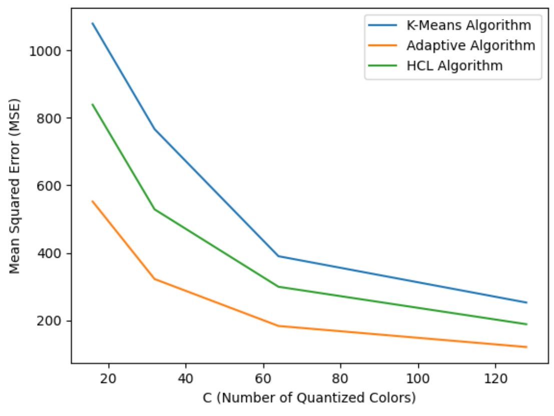
(a) House



(b) Jellybeans



(c) Baboon



(d) Peppers

Figure 5.2: Mean Squared Error (MSE) Plots

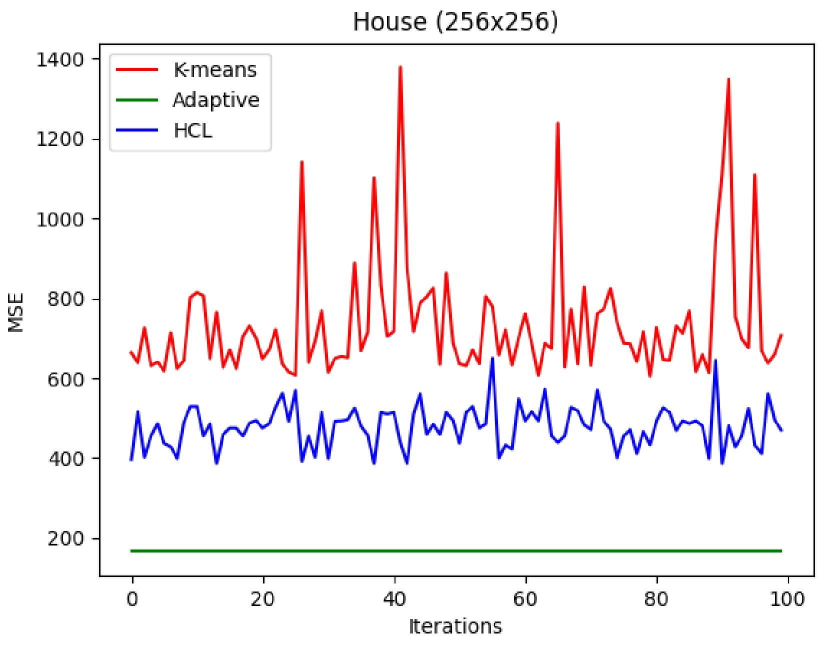
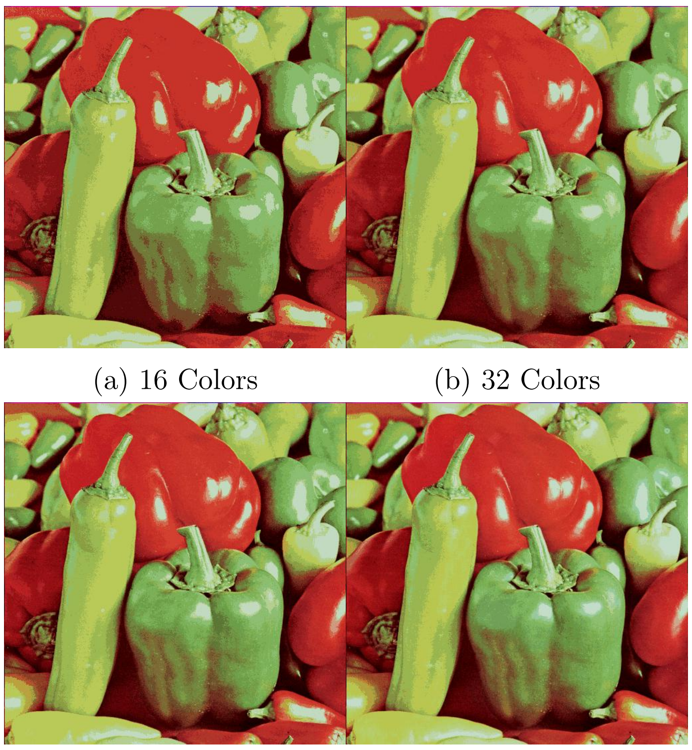
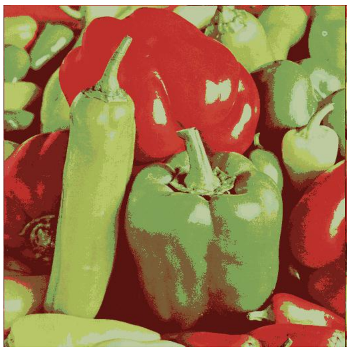


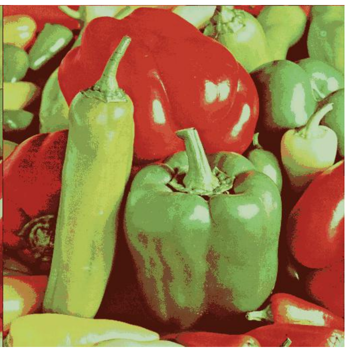
Figure 5.3: Change in MSE over multiple Iterations (Depicting the dependence on initial conditions)

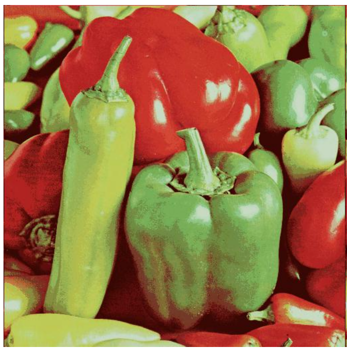

(a) 16 Colors (b) 32 Colors

(c) 64 Colors (d) 128 Colors

Figure 5.4: K-Means (peppers)


(a) 16 Colors


(b) 32 Colors


(c) 64 Colors

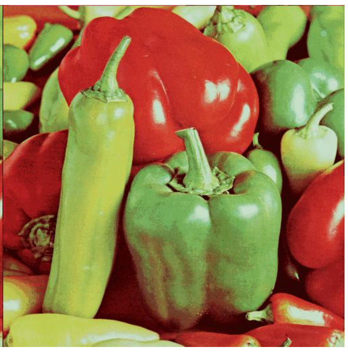
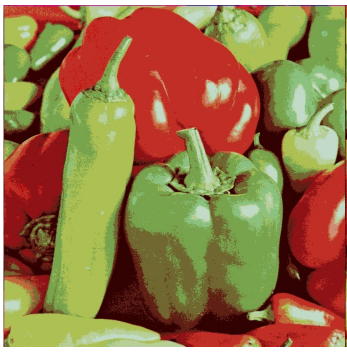
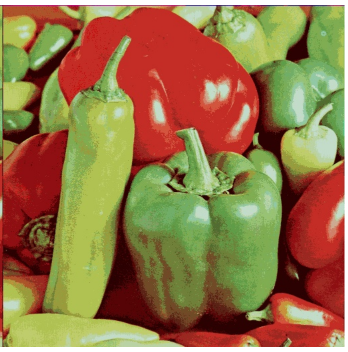
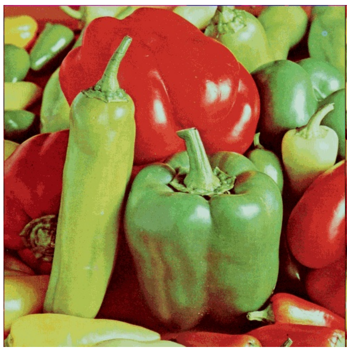

(d) 128 Colors

Figure 5.5: HCL (peppers)


(a) 16 Colors


(b) 32 Colors


(c) 64 Colors

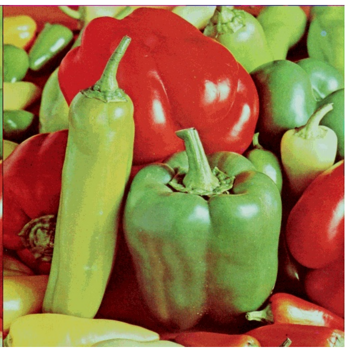

(d) 128 Colors

Figure 5.6: Adaptive (peppers)

Chapter 6

Conclusion

In Table 5.1 and Figure 5.1, all the runtimes are shown. Runtime of HCL is observed to be maximum. This is because in HCL, after each splitting, it uses Competitive Learning (CL) to cluster the image pixels. Runtime of Adaptive algorithm is almost independent of the number of quantized colors. This is because most of the time is spent on preprocessing and mapping of pixels to the quantized colors. The actual time taken by the algorithm is significantly less than those times. K-means algorithm takes the least time when the number of quantized colors is 16, but it increases at a faster rate than the runtime of adaptive algorithm and takes more time than the adaptive algorithm when the number of quantized colors increases.

Table 5.2 and Figure 5.2 show MSEs for all the images in the dataset. Adaptive algorithm performs the best. Although K-means takes lesser time than the HCL algorithm, it gives larger MSE than the HCL algorithm in every case.

The dependency of the algorithms on initial condition is observed using Figure 5.3. K-means algorithm is heavily dependent on the initial conditions. The best and worst case differ by almost a factor of 2 showing that the algorithm converges to a local solution. The slight change in MSE distribution for HCL algorithm demonstrates that this algorithm converges to a solution near the global optimum. Adaptive algorithm is completely independent of the initial conditions.

Chapter 7

References

- An adaptive clustering algorithm for color quantization, Ing-Sheen Hsieh, Kuo-Chin Fan
- A comparison of clustering algorithms applied to color image quantization, P. Scheunders
- https://towardsdatascience.com/color-palette-extraction-with-k-means-clustering-machine-learn ing-from-scratch-part-iv-55e807407e53