THE EFFECTS OF LEARNING RATE SCHEDULES ON COLOR QUANTIZATION

by

Garrett Brown

A thesis presented to the Department of Computer Science
and the Graduate School of the University of Central Arkansas
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Conway, Arkansas
May 2020

TO THE OFFICE OF GRADUATE STUDIES:

The members of the Committee approve the thesis of

Garrett Brown presented on March 31, 2020.

M. Emre Celebi Digitally signed by M. Emre Celebi Date: 2020.04.09 17:47:02 -05'00'

M. Emre Celebi, Ph.D., Committee Chairperson

Ahmad Patooghy Digitally signed by Ahmad Patooghy Date: 2020.04.09 17:53:10 -05'00'

Ahmad Patooghy, Ph.D.

Mahmut Karakaya Digitally signed by Mahmut Karakaya Date: 2020.04.09 19:52:03 -05'00'

Mahmut Karakaya, Ph.D.

PERMISSION

Title The Effects of Learning Rate Schedules on Color Quantization

Department Computer Science

Degree Master of Science

In presenting this thesis/dissertation in partial fulfillment of the requirements for a graduate degree from the University of Central Arkansas, I agree that the Library of this University shall make it freely available for inspections. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis/dissertation work, or, in the professor's absence, by the Chair of the Department or the Dean of the Graduate School. It is understood that due recognition shall be given to me and to the University of Central Arkansas in any scholarly use which may be made of any material in my thesis/dissertation.

Garrett Brown

March 31, 2020

ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Dr. Emre Celebi, professor and chair of the Department of Computer Science at University of Central Arkansas. He was always willing to discuss any issues or concerns I had. Dr. Celebi consistently guided me throughout the project to ensure I was going in the right direction.

I would also like to acknowledge the rest of the faculty of the Department of Computer Science at University of Central Arkansas for all the guidance throughout my graduate and undergraduate studies. They were able to provide me an opportunity to discover and pursue my passion for Computer Science.

Finally, I would like to express my gratitude to my family and wife for supporting me through all of my years of study. It would not have been possible without them.

VITA

Garrett Brown was born in Ventura, California. He attended elementary, middle, and high school in Vilonia, Arkansas and graduated with honors in 2014. He attended University of Central Arkansas and graduated from UCA Magna Cum Laude with a Bachelor of Science in Computer Science and a minor in Mathematics in August 2018.

ABSTRACT

Color Quantization is a technique to reduce the number of distinct colors in a digital color image and is an important operation in image processing and computer graphics. In this thesis, we improve upon a recently proposed color quantization method. The proposed method utilizes an efficient algorithm to determine the initial color palette and quasirandom sampling to scramble the image deterministically. An online version of the k-means clustering algorithm is then used to fine-tune the initial palette. We investigate multiple learning rate schedules to control the convergence of online k-means. The resulting color quantization method is trained on 20 images and tested on 8 popular images. The results demonstrate that while the proposed method is competitive with state-of-the-art methods with respect to quality, it is faster.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

A true color image is typically comprised of pixels represented by three 1-byte color components: red, green, and blue. This means that 3 bytes are needed to store a color pixel and each color pixel can assume a value out of 256^3 color combinations. Color quantization (CQ) is an image processing method that reduces the number of unique colors in an image, allowing the image to be stored with fewer colors while maintaining the fidelity of the image as much as possible. While CQ is no longer required due to the advancement of hardware technology, it is still beneficial for various computer graphics and image processing applications. Applications of CQ in the aforementioned fields include color texture analysis, compression, watermarking, and others (Celebi 2011).

Quantization Methods

A large number of CQ methods have evolved over the past four decades, but virtually any CQ method belongs to one of two groups: preclustering methods and postclustering methods (Brun & Trémeau 2003). The former methods discover clusters recursively either top-down or bottom-up, thus leading to a hierarchy in the clustering (Jain, Murty & Flynn 1999). Postclustering methods, on the other hand, find all the clusters at once via partitioning and do not impose any structure onto the data. The drawback of postclustering is that it is very time consuming even though it produces better results because it improves upon the initial color palette iteratively.

Color Quantization via Data Clustering

Because a pixel is made up of a red, a green, and a blue component, CQ can be interpreted as a three-dimensional clustering problem where the goal is to determine the clusters that best represent the colors in an image (Celebi 2009). CQ methods can be

broken into two main phases: palette design and mapping. In the palette design phase, the algorithm generates a reduced color palette with *k* colors, where *k* is a user-defined integer typically ranging from 8 to 256. In the second phase, each pixel in the original image is mapped to one of the colors in the reduced palette.

Metaheuristics and Color Quantization

As the techniques for CQ have evolved, researchers have started to borrow ideas from other fields such as machine learning. One such result of this is the combination of metaheuristics and preclustering or postclustering methods. Metaheuristics based CQ methods include a hybrid method combining k-means clustering and self-adaptive differential evolution (Su & Hu 2013), artificial bee colony optimization (Ozturk, Hancer, & Karaboga 2014), ant-tree algorithm (Pérez-Delgado 2015), combining fuzzy c-means clustering and artificial fish swarm algorithm (El-Said 2015), combining k-means with the harmony search algorithm (Khaled, Abdel-Bader, & Yasein 2016), artificial ants and fireflies (Pérez-Delgado 2018), binary splitting and ant-tree algorithm (Pérez-Delgado & Gallego 2018), artificial bee colony algorithm combined with ant-tree algorithm (Pérez-Delgado 2019) shuffled-frog leaping algorithm (Pérez-Delgado 2019), greedy orthogonal bi-partitioning and ant-tree (Pérez-Delgado & Gallego 2019) and particle swarm and artificial ants (Pérez-Delgado 2020); these methods present CQ as a global optimization problem that can be solved with nature- or physics-inspired metaheuristics. While these methods can be more powerful than the traditional preclustering/postclustering methods because of their ability to optimize complex objective functions. Unfortunately, these methods are typically randomized and computationally intensive, with multiple parameters that are hard to fine tune for each image.

CHAPTER 2 RELATED WORK

Many CQ techniques exist in the literature, such as median-cut (Heckbert 1982), popularity (Heckbert 1982), modified popularity (Braudaway 1987), greedy orthogonoal bipartitioning (Wu 1991), octree (Gervautz & Purgathofer 1988), center-cut (Joy & Xiang 1993), variance-based method (Wan, Prusinkiewicz, &Wong 1990), modified maximin (Xiang 1997), radius-weighted mean-cut (Yang & Lin 1996), self-organizing map (Dekker 1994), Cheng and Yang (Cheng & Yang 2001), split and merge (Brun, & Mokhtari 2000), weighted sort-means (Celebi 2009), modified weighted sort-means (Celebi 2011), fuzzy c-means (Wen & Celebi 2011), adaptive distributing units (Celebi, Hwang, & Wen 2014), variance-cut (Celebi, Wen, & Hwang 2015), and MacQueen's k-means (Thompson, Celebi, & Buck 2020).

Heckbert (1982) proposed two CQ methods: popularity and median-cut. Popularity builds a 16 x 16 x 16 color histogram and derives the reduced color palette from the *k* most frequent colors in the histogram. Median-cut builds a 32 x 32 x 32 color histogram whose volume is recursively split along the longest axis. The histogram is split until there are *k* boxes for the color palette.

Braudaway (1987) introduced the modified popularity method that builds a 2^R x 2^R x 2^R color histogram with *R* bits per channel. The most frequent color is taken as the first color in the reduced palette and then the frequency of each of the remaining colors is reduced. This procedure is repeated until a total of *k* colors is chosen.

Gervautz and Purgathofer (1988) discussed the octree method. It builds an octree that is representative of the color distribution of the image. Then it merges adjacent

colors with the fewest pixels to the nearest cluster form the bottom up until it has *k* colors.

Wan, Prusinkiewicz, and Wong (1990) introduced the variance-based method. This method is like the median-cut but the box with the greatest error is split along the least weighted sum of projected variances axis.

Wu (1991) presented the greedy orthogonal bipartitioning procedure. It works like the variance-based method but it splits on the axis that minimizes the sum of variance.

Joy and Xian's (1993) center-cut method is similar to the median-cut but splits the box with the greatest range on the average point in the box.

Dekker (1994) provided the self-organizing map scheme. It has a self-organizing map of *k* neurons and a random subset of the pixels are used for the training phase. The final weights from the training phase are used as the color palette.

Yang and Lin (1996) proposed the radius-weighted mean-cut method. This is almost exactly like the variance-based method but it splits on the vector from the origin to the radius-weighted mean.

Xiang (1997) introduced a variant of the maximin method (Gonzalez 1985). The first color is chosen randomly and each subsequent color is chosen as the color with the greatest minimum weighted distance to the previous colors. The palette colors are then calculated as the mean of the colors assigned to these initial colors.

Brun and Mokhtari (2000) developed the split and merge method. This method partitions the color space into *B* partitions that are represented on an adjacency graph. Then each cluster is merged by joining the clusters with the least amount of error.

Cheng and Yang's (2001) self-named method works like the variance based-method except it is split along a specific line determined by the mean color and the color farthest from it.

Celebi (2009, 2011) introduced the weighted sort-means clustering method. It is an adaptation of the traditional k-means clustering algorithm that involves sample weighting, data reduction, and accelerated nearest neighbor search.

Wen and Celebi (2011) proposed the fuzzy c-means clustering that modifies the k-means algorithm so that points can belong to more than one cluster. The goal is to generate an optimal fuzzy c-partition by minimizing a fuzzy objective function.

Celebi, Hwang, and Wen (2014) brought forth the adaptive distributing units (ADU) method, which is an adaption of Uchiyama and Arbib's clustering algorithm (1994). It is a competitive learning algorithm where points compete to represent the input. The winner is moved closer to the input by designated learning rate. The method starts with a unit that is represented by the center of all the inputs and then new units are generated by splitting existing units until there are K units.

Finally, Thompson, Celebi, and Buck (2020) proposed a modified version of MacQueen's (1967) online k-means algorithm with a square-root learning rate schedule.

This thesis expands upon the recent work by Thompson, Celebi, and Buck to investigate how alternative learning rate schedules (LRS's) and parameters affect the results of a CQ method based on MacQueen's k-means algorithm. We propose a new learning rate schedule that is very competitive with Thompson, Celebi, and Buck's CQ method in terms of quantization quality while being faster.

CHAPTER 3 PROPOSED COLOR QUANTIZATION METHOD

This chapter describes the proposed CQ method in detail. First, we present an efficient and deterministic initialization method that is used for determining the initial color palette with *k* colors. Then, we describe a quasirandom sampling method that is utilized for selecting points to be presented to MacQueen's k-means clustering algorithm. Finally, we explain the clustering algorithm itself.

Initialization

Before we can use a postclustering method, we need to determine the initial color palette, which has *k* colors (*k* is a user-defined integer). The problem of devising an initial color palette of size *k* in a CQ application is essentially the same problem as selecting the initial *k* cluster centers in a partitional clustering application (Celebi 2015).

An often-used approach for determining the initial *k* cluster centers is to select these points uniformly at random from the entire data set. Unfortunately, the k-means clustering algorithm is rather sensitive to initialization which can lead to negative consequences such as longer convergence time, empty clusters, and a larger likelihood of the algorithm getting trapped at a local minimum (Celebi, Kingravi, & Vela 2013). Gonzalez (1985) proposed a better initialization method, named the maximin initialization method, that selects the first center arbitrarily and each of the remaining centers is chosen as the point with the greatest minimum distance to the previously selected centers.

For this algorithm, described in Table 1, the first center is typically chosen uniformly at random from data set. However, for the purposes of this thesis, the average red, green, and blue color across the input image was used as the first center. This allows

the entire initialization process to become deterministic as long as a deterministic tie-breaking strategy is utilized when finding nearest point-to-center distances. An often-used strategy, and the one used in this work, is having the smallest index determine the tiebreaker. The remaining $k - 1$ centers are chosen such that the ith center ($i \in \{2, \ldots, k\}$) is the point with the greatest minimum distance to the nearest of the previously selected ($i - 1$) centers.

Table 1. *Maximin Initialization Algorithm*

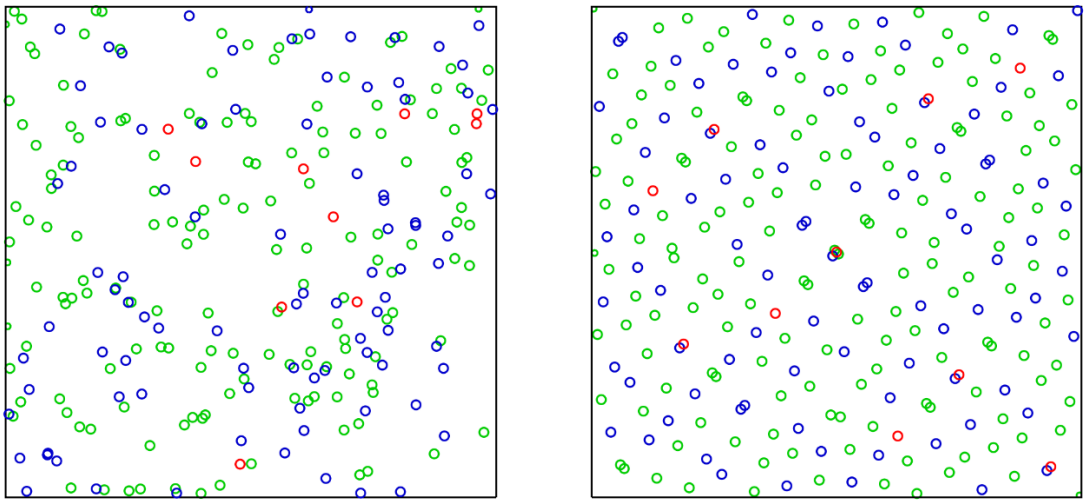
Step	Description
1	Select the first center c_1 as the mean data point.
2	Select the next center c_i as the point with the greatest minimum distance to the previously selected ($i - 1$) centers.
3	Repeat Step 2 until all k centers have been determined.

Sampling

Online clustering algorithms like MacQueen's tend to be more adaptive so the order in which the points are presented has significance (Thompson, Celebi, & Buck 2020). If the points are presented in raster order, then the algorithm will learn of many of the same points due to the nature of image data and the accuracy of learning will decrease. If the points were sampled *pseudorandomly*, some clumping would occur on among the selected points, which leads to potential bias in the learning procedure. Pseudorandom sampling could also lead to vastly different clustering results in each run. The Sobol' sequence (Bratley & Fox 1988) presents a way to sample the data *quasirandomly* to avoid the previously mentioned bias. The Sobol' sequence samples

more uniformly, see Figure 1, and in a way that is deterministic, so the online algorithm becomes order independent.

Figure 1. *Comparison of pseudorandom (a) and quasirandom (b) (https://en.wikipedia.org/wiki/Sobol_sequence)*



a) Pseudorandom Sampling b) Quasirandom Sampling

Clustering

While Lloyd's (1982) *batch k-means* algorithm (Linde *et al.*, 1980) is the most frequently used k-means variant (Wu *et al.* 2008), there is another variant of k-means, often referred to as the *online k-means* (MacQueen 1967). This algorithm starts with *k* centers, where *k* denotes the number of clusters (or, number of colors in a CQ application). With the previously discussed maximin initialization algorithm, the *k* centers have already been initialized as opposed to selecting the first *k* points as the initial centers like MacQueen suggests in his classical work. The algorithm utilizes a randomly sampled point *x* and assigns it to the nearest cluster, which is then updated to include this

new point. This update includes increasing the size of the cluster as well as adjusting the center as follows

$$c^{(t+1)} = c^{(t)} + \eta(t)(x^{(t)} - c^{(t)}),$$

where $x^{(t)}$ is the current presented point, $c^{(t)}$ is the current center of the nearest cluster, $c^{(t+1)}$ is the new center of the nearest cluster that now includes $x^{(t)}$, and $\eta(t)$ is the learning rate bounded in (0,1]. The learning rate is selected to satisfy the Robbins–Monro conditions

$$\lim_{t\to\infty} \eta(t) = 0,$$

$$\sum_{t=1}^{\infty} \eta(t) = \infty,$$

$$\sum_{t=1}^{\infty} \eta(t)^2 < \infty.$$

These conditions ensure that η decreases slow enough to avoid premature convergence, but not so slow to be overly influenced by noise (Robbins & Monro 1951). The rate allows for the algorithm to learn in a way that the quantization distortion is decreased. MacQueen's algorithm then continues through each sampled point as determined by the Sobol' sequence, updating the nearest center after each presentation.

Learning Rate Schedules

As mentioned earlier, the LRS is important for the clustering algorithm to properly and efficiently converge. This work utilizes three different LRS's and two variants bringing the total to five. These schedules are detailed below.

Constant

The constant learning rate is defined by $\eta(t) = c$ with $0 < c \leq 1$. With this schedule, each color in the quantized palette is a result of the exponentially decaying average of the inputs that were assigned to the corresponding cluster (Fritzke 1997). The

value of *c* is inversely related to the number of inputs because as more points are presented the average of the inputs decay faster. Therefore, the rate *c* should be much smaller than 1 (e.g., 0.001) because an image is typically made up of hundreds of thousands of pixels.

Hyperbolic

The hyperbolic learning rate is the most popular LRS in the stochastic approximation literature (Robbins & Monro 1951). It is given by $\eta(t) = \frac{1}{(1+t)^p}$, where *t* indicates the cardinality of the cluster before including the new point and *p* is the exponent bounded in the interval (0.5, 1] to ensure convergence. When $p = 1$, the hyperbolic learning becomes the rate that MacQueen originally used for his algorithm, which again is the most popular rate for stochastic approximation. This rate provides a running average of the presented data. When $p < 1$, this LRS leads to an increasing weight sequence thus giving more weight to the latest data (Yair, Zeger, & Gersho 1992). And, when *p* is at its lower bound of 0.5, it tends to lead to faster convergence for finite data sets (Darken & Moody 1990).

Linear

The linear decay LRS is given by $\eta(t) = c(1 - t/T)$, where $0 < c \leq 1$ and *T* is the total number of points to be presented to the learning algorithm. When *t* is very small compared to *T*, the learning rate is very slow and can be compared to the constant LRS. On the opposite end, when *t* approaches *T*, the learning rate rapidly approaches zero. So, there is some intermediate point *t*, where the schedule has the best point of learning (Yair, Zeger, & Gersho 1992). With a smaller constant (e.g., $c = 0.01$), the LRS approaches to

that intermediate point more slowly and learns as much as possible before ceasing to learn.

Parametric Hyperbolic

The parametric hyperbolic LRS is defined by $\eta(t) = \frac{k}{(k+1+t)^p}$, which makes it similar to the original hyperbolic LRS with the exception that the parametric version utilizes the number of clusters (*k*) in its formulation. Recall that *k* is actually a parameter of the clustering algorithm rather than a parameter of the LRS. The use of *k* allows the LRS to adjust the learning rate proportionally to *k* because when *k* is large, the average number of presentations per cluster is small, so the learning rate should be larger (otherwise, the algorithm will not have sufficient opportunity to learn each center) (Wu & Yang, 2006). Just like the hyperbolic LRS, the parametric hyperbolic LRS has the exponent *p*, which is bounded in the interval (0.5, 1].

Alternative Parametric Hyperbolic

The alternative parametric hyperbolic schedule is defined by $\eta(t) = \frac{c}{(fT+1+t)^p}$. It is another modified version of the hyperbolic LRS and is the schedule with the most adjustable parameters. Just like the hyperbolic LRS, it has the exponent *p* bounded in the interval (0.5, 1]. The parameter *f* is a fractional multiplier that adjusts the total number of points to be presented. Finally, there is a positive parameter *c* that scales the overall learning rate.

CHAPTER 4 EXPERIMENTAL RESULTS AND DISCUSSION

This chapter explains the experimental procedure that was followed and then details the experimental results. First the training and test data sets and the performance measures are described. Then, all the constant and variable parameters involved in the experiment are discussed. Finally, the results of the experiments are presented and their results are discussed.

Experimental Setup

The program first reads the image data from a Portable Pixmap, or PPM, image file and gets the mean image color to be used for the first cluster center. These PPM images have header data that contains the image height and width, the maximum brightness value which determines the number of bits-per-pixel for each channel, and the image format. The algorithm was trained with 20 PPM images and the best parameters were tested on 8 images commonly used in the CQ literature as seen in Table 2. Lenna, Peppers and Baboon are from the USC-SIPI Image Database (http://sipi.usc.edu/database); Parrots and Motocross are from the Kodak Lossless True Color Image Suite (http://r0k.us/graphics/kodak/); and Pills, Fish, and Goldhill are by Karel de Gendre, Luiz Velho, and Lee Crocker respectively. Then the LRS is selected based on a parameter passed to the program. The cluster centers are initialized with the maximin method. Once the MacQueen's clustering algorithm is completed, the final *k* centers are taken as the reduced color palette. This color palette is then used to map the input image by determining the nearest color to each input pixel and replacing that pixel with that palette color. The resulting image is then comprised of *k* colors as opposed to

the initial number of colors it had. The program then generates the output PPM file based on the quantized output image. Finally, to determine the distortion, the input image and the output image are compared using the Mean Squared Error measure that is described in the next section.

Table 2. *Test image set*

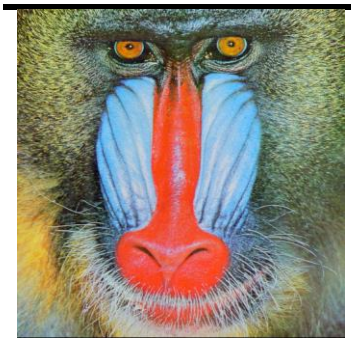
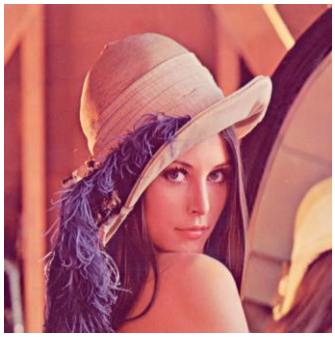
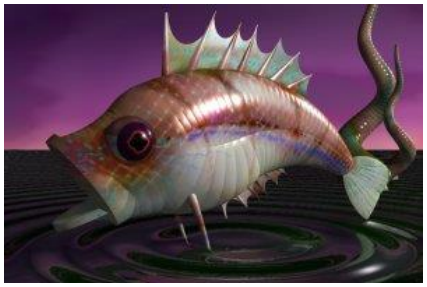
Image	File Name	Size	Number of Colors
	Baboon	512 x 512	230,427
	Lenna	512 x 512	148,279
	Peppers	512 x 512	183,525
	Fish	300 x 200	63,558

Table 2. *(continued)*

	Goldhill	720 x 576	30,966
	Motocross	768 x 512	63,558
	Parrots	768 x 512	72,079
	Pills	800 x 519	206,609

Performance Measures

Time

The computation efficiency of the initialization and clustering algorithm were measured using the CPU time. The time was measured using the *high_resolution_clock* class of the *chrono* library that exists in the *std* namespace for C++. This clock type has a resolution of a microsecond. The elapsed time was measured by using the *now* function of the class, which gets the current time, before the initialization happens and after the clustering is completed then taking the difference of the two instances. The measurements were then converted to milliseconds for readability and comparison.

Mean Squared Error

A popular image quality metric is the Mean Squared Error (MSE). It is calculated by taking the average of the squared differences of the corresponding pixels from two images. This method for evaluating the quality is often used in CQ literature due to its simplicity and clear meaning. It is computed as:

$$MSE(X, \hat{X}) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} ||X(h,w) - \hat{X}(h, w)||^2,$$

where X and $\hat{X}$ represent, respectively, the H × W original and quantized images in the RGB color space.

Experimental Parameters

The variable parameters for MacQueen's algorithm were represented as follows: the number of colors k = {32, 64, 128, 256}; presentation factor f = {0.25, 0.5, 0.75, 1}; parametric fraction pf = {0.25, 0.5, 0.75}; and parametric constant pc = {1, 2, 3}. Every time the program ran, the MSE was captured for each LRS and the parameters involved. This MSE data was used determine optimal parameter configurations for each of the

LRS's. The mean and standard deviation of CPU time was also captured over 100 runs for 3 of the 8 test images (Baboon, Lenna, and Peppers) as their sizes were identical (512 x 512), allowing easier comparisons. This program was implemented with the C++ programming language, compiled with the GNU g++ compiler version 8.1.0 and executed on a 2.20GHz Intel Core i7-8750H CPU.

Discussion

The experimental results are given in Tables 3–7. Tables 3–6 contain the MSE for each LRS, and Table 7 shows the CPU time, in milliseconds, for the *k* values tested. Figures 1–3 show a cropped section of one of the original images and the same cropped outputs from different LRS's. Note that, in these tables, we also compare the MacQueen's k-means algorithm with the proposed LRS's to the batch k-means algorithm (BKM). For comparisons against other common CQ methods, please refer to Thompson, Celebi, and Buck (2020). Note that in these tables, the row labeled as 'Hyper' corresponds to the MacQueen's k-means algorithm with the square-root hyperbolic LRS (the MKM_q algorithm in Thompson, Celebi, & Buck) In the initial experimentation, the MSE values were normalized using min-max normalization in the [0, 100] interval across each training image and their averages and standard deviations were computed to allow better comparison across each LRS.

Table 3. MSE for $k = 32$ colors

	Image			
LRS	Baboon	Lenna	Peppers	Fish
Const	386.0	134.4	273.2	147.9
Hyper	375.6	131.4	258.4	148.8
Linear	383.9	134.7	272.3	147.4
Phyper	375.7	131.5	259.5	144.1
AltPhyper	376.6	131.8	258.4	146.2
BKM	374.2	130.8	248.7	142.6

	Image			
LRS	Goldhill	Motocross	Parrots	Pills
Const	147.2	203.9	236.2	202.8
Hyper	144.5	194.5	241.4	199.1
Linear	146.8	207.5	234.3	201.8
Phyper	143.9	188.5	230.7	197.7
AltPhyper	144.9	199.7	232.2	202.0
BKM	143.8	197.5	230.7	198.4

Table 4. MSE for $k = 64$ colors

	Image			
LRS	Baboon	Lenna	Peppers	Fish
Const	242.1	77.3	154.6	95.8
Hyper	236.3	75.5	149.2	93.1
Linear	240.6	77.7	156.0	97.7
Phyper	238.0	75.7	147.8	92.4
AltPhyper	238.0	75.8	150.4	95.5
BKM	234.3	74.7	148.1	90.2

	Image			
LRS	Goldhill	Motocross	Parrots	Pills
Const	87.1	117.8	131.3	117.0
Hyper	83.5	116.9	127.4	112.2
Linear	88.1	119.7	132.8	115.9
Phyper	83.9	109.2	127.0	112.0
AltPhyper	85.2	115.7	130.6	113.4
BKM	83.0	115.0	129.5	111.1

Table 5. MSE for $k = 128$ colors

	Image			
LRS	Baboon	Lenna	Peppers	Fish
Const	155.2	49.7	93.8	62.9
Hyper	152.2	48.0	89.6	59.4
Linear	154.5	50.2	94.2	63.0
Phyper	159.0	49.7	92.4	62.9
AltPhyper	153.3	48.5	91.3	60.9
BKM	149.3	46.8	87.7	57.3

	Image			
LRS	Goldhill	Motocross	Parrots	Pills
Const	57.1	76.0	78.0	71.8
Hyper	53.5	72.9	76.1	67.5
Linear	57.7	77.2	78.9	71.8
Phyper	54.2	65.3	74.9	68.5
AltPhyper	55.0	75.6	76.5	68.8
BKM	52.0	68.0	73.2	66.3

Table 6. MSE for $k = 256$ colors

LRS	Image: Baboon	Lenna	Peppers	Fish
Const	100.9	33.3	62.3	40.1
Hyper	97.8	31.5	57.7	36.4
Linear	100.9	33.4	63.0	40.6
Phyper	109.6	34.4	63.1	45.5
AltPhyper	100.0	32.7	60.9	39.2
BKM	95.6	30.3	55.0	34.8

LRS	Image: Goldhill	Motocross	Parrots	Pills
Const	38.5	48.7	47.6	46.1
Hyper	35.8	44.8	44.5	42.5
Linear	38.8	48.8	48.3	46.4
Phyper	38.0	44.3	47.5	45.9
AltPhyper	38.1	48.1	47.3	44.7
BKM	34.2	42.9	44.3	41.0

Table 7. CPU Time in Milliseconds

		Baboon		Lenna		Peppers	
LRS	Colors	Mean	Stdev	Mean	Stdev	Mean	Stdev
Constant	32	154.48	7.36	152.72	6.52	152.57	8.87
	64	281.20	7.05	278.97	8.05	279.97	8.74
	128	531.77	10.68	528.94	10.94	528.88	11.15
	256	1020.68	17.13	1020.89	15.15	1015.69	11.82
Hyper	32	171.38	7.04	170.92	14.36	168.51	7.30
	64	296.60	6.79	292.53	10.12	293.04	7.60
	128	546.66	8.16	541.93	11.70	546.61	36.96
	256	1040.13	9.79	1038.73	10.89	1035.57	11.18
Linear	32	152.92	7.63	151.08	7.49	148.72	7.49
	64	280.99	6.29	277.93	7.04	278.78	6.85
	128	528.33	7.77	524.63	11.64	525.02	11.36
	256	1018.68	13.32	1016.22	11.60	1014.48	11.24
Phyper	32	170.44	6.18	166.38	7.69	166.07	7.51
	64	299.19	6.72	297.11	19.95	294.65	12.75
	128	545.18	8.07	541.91	8.49	543.11	8.78
	256	1044.39	14.18	1038.29	10.17	1042.03	10.40
AltPhyper	32	136.16	7.08	138.87	17.65	136.87	12.88
	64	263.07	29.69	258.28	27.33	263.20	32.26
	128	496.63	29.91	519.70	71.38	476.88	14.99
	256	920.27	12.36	917.09	12.49	916.71	12.60
BKM	32	4452		4357		2646	
	64	14325		9334		6333	
	128	27342		26525		33274	
	256	40661		33235		30559	

Figure 1. *Baboon Output Images (k = 32)*

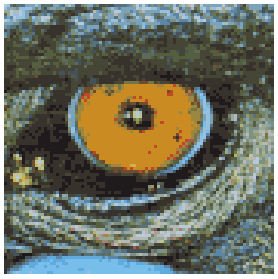
a) Original

b) Constant Output

c) Hyper Output

d) Linear Output

e) PHyper Output

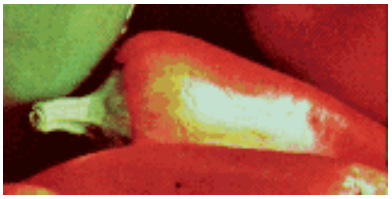


f) AltPhyper Output

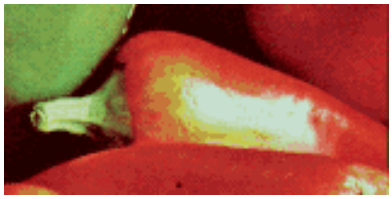
g) BKM Output

Figure 2. *Peppers Output Images (k = 64)*

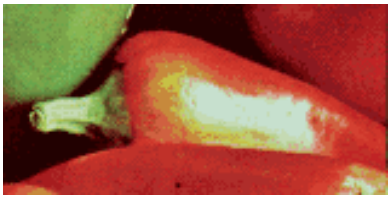
a) Original



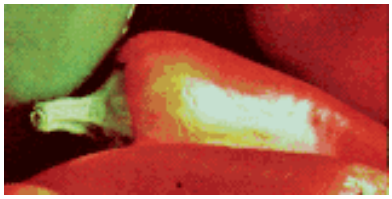
b) Constant Output



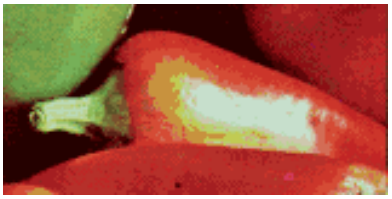
c) Hyper Output



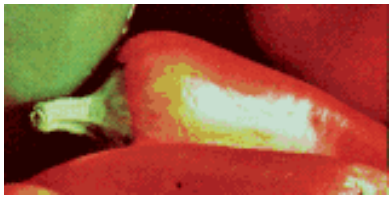
d) Linear Output



e) PHyper Output



f) AltPhyper Output



g) BKM Output

Figure 3. *Pills Output Images (k = 128)*

a) Original

b) Constant Output

c) Hyper Output

d) Linear Output

e) PHyper Output

f) AltPhyper Output

g) BKM Output

Constant

The constant LRS, as well as every other LRS, typically performed better with a greater presentation factor. As more of the image was presented, the algorithm was able to learn more from it and provide a more accurate color quantized image, as shown in Figure 4. For constant LRS specifically, the value of c = 0.03 yielded the best results. It had the lowest average MSE as well as the lowest standard deviation across all the images.

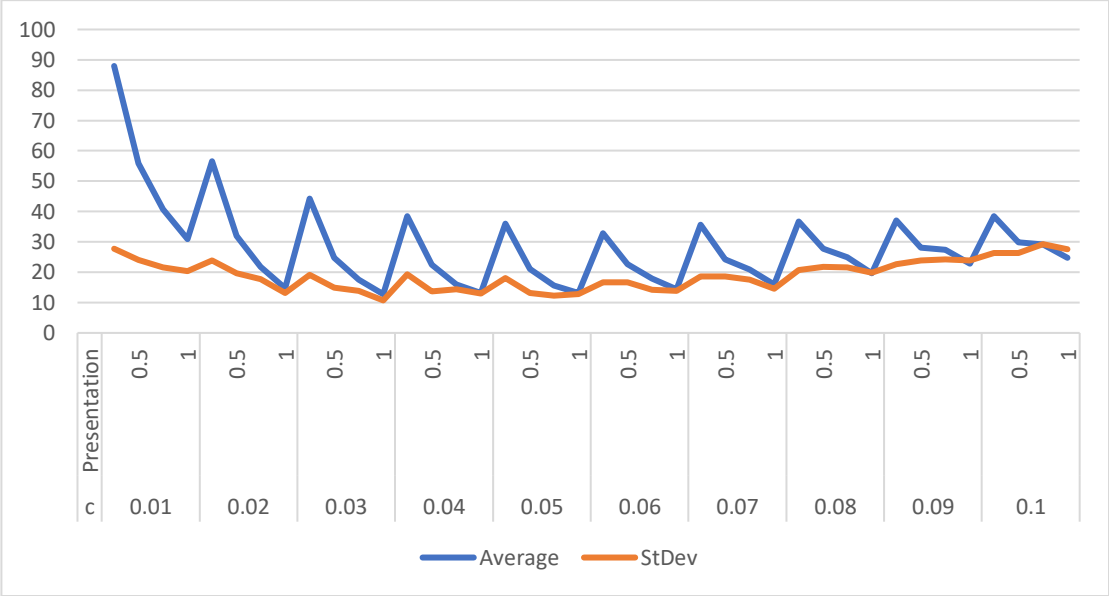


Figure 4. *Average and Standard Deviation for Constant LRS training*

When using the optimized parameters on the 8 test images, the constant LRS produced many of the higher MSE values across all *k* values. The MSE was higher for lower number of colors because there were fewer clusters for the points to be assigned, meaning that each cluster was potentially larger. This caused the exponentially decaying average of the inputs to decay faster than it did for higher number of colors. At the higher number of colors, fewer points were added to each cluster so the average did not decay as

fast. Even though the MSE was still worse compared to other LRS's; the higher color number of colors allowed for the constant LRS to be relatively more competitive.

Hyperbolic

This LRS gave the best average MSE for $p = 0.5$, which is consistent with what has been observed in the literature (Darken & Moody 1990) (Wu & Yang 2006) (Thompson, Celebi, & Buck, 2020). The LRS with $p = 0.5$ also produced very low standard deviations, further indicating that the LRS was consistently the best for this parameter value. As the parameter p approached 1, the same parameter value that the MacQueen originally used, the average MSE continued to increase, see Figure 5, showing that the most popular rate in the stochastic approximation literature is not as good in practice as it is in theory.

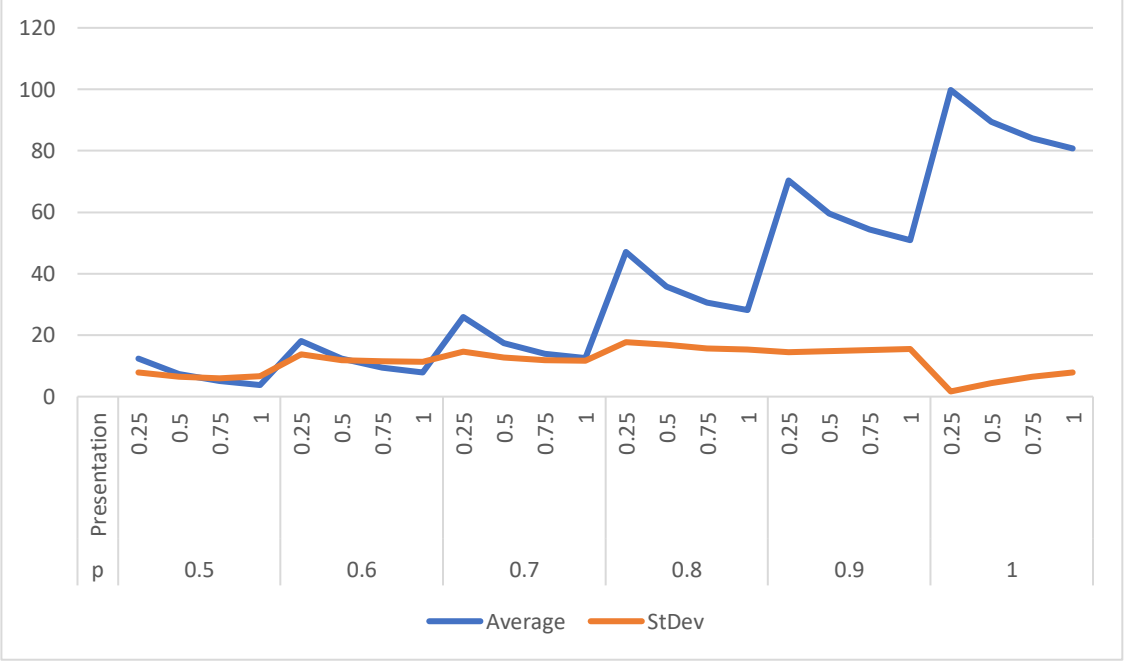


Figure 5. Average and Standard Deviation for Hyperbolic LRS training

Linear

This LRS ended up being very similar to the Constant LRS, see Figure 6. This is likely due to *t* being very small compared to the total points presented as mentioned in the previous chapter. So, until a cluster had enough points in it, the linear LRS basically acted like a constant one. There is some difference between the two LRS's, however, as c = 0.05 ended up being the parameter giving the lowest average MSE rather than the c = 0.03 for the constant LRS.

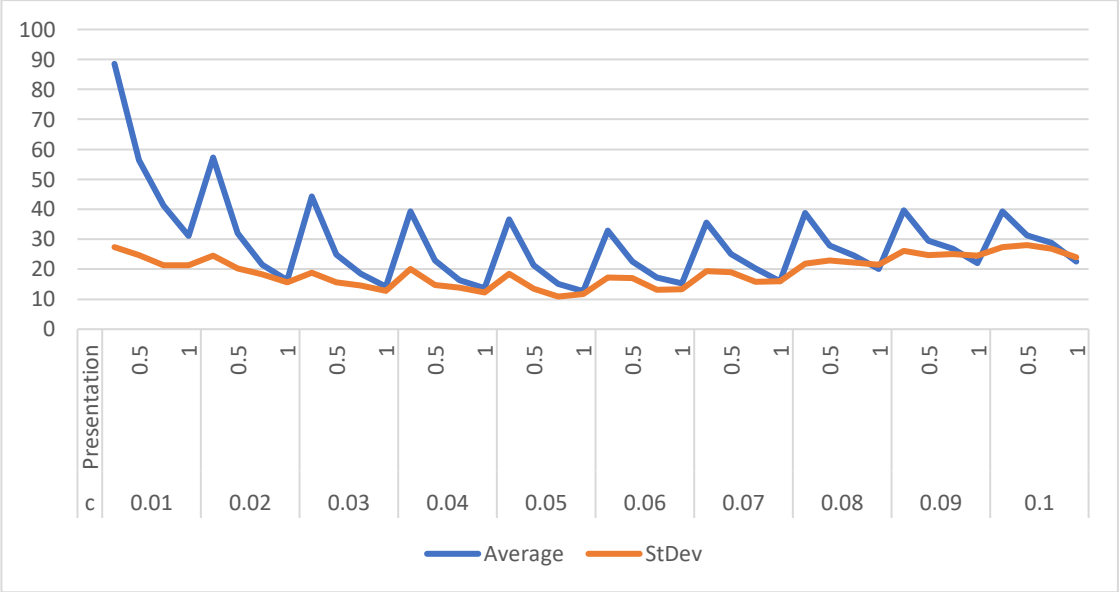


Figure 6. *Average and Standard Deviation for Linear LRS training*

When utilizing the best c parameter on the test images, many of the MSE values were very close to those given by the constant LRS. However, many of the configurations using this LRS had the worst MSE across all investigated schedules. As mentioned earlier, when there are few points, the linear LRS behaves like the constant LRS. By the time the t/T portion of the equation started to factor in, it was likely that there was very little more learning that could be done using this schedule because of how quickly the constant LRS decays.

Parametric Hyperbolic

The parametric hyperbolic LRS, shown in Figure 7, although very similar to the conventional hyperbolic LRS, did not behave in the same way. As the exponent *p* is increased, the average MSE decreased (up to a certain point) rather than increase like hyperbolic does. The standard deviations for this LRS also seemed to be the highest across all of the LRS's.

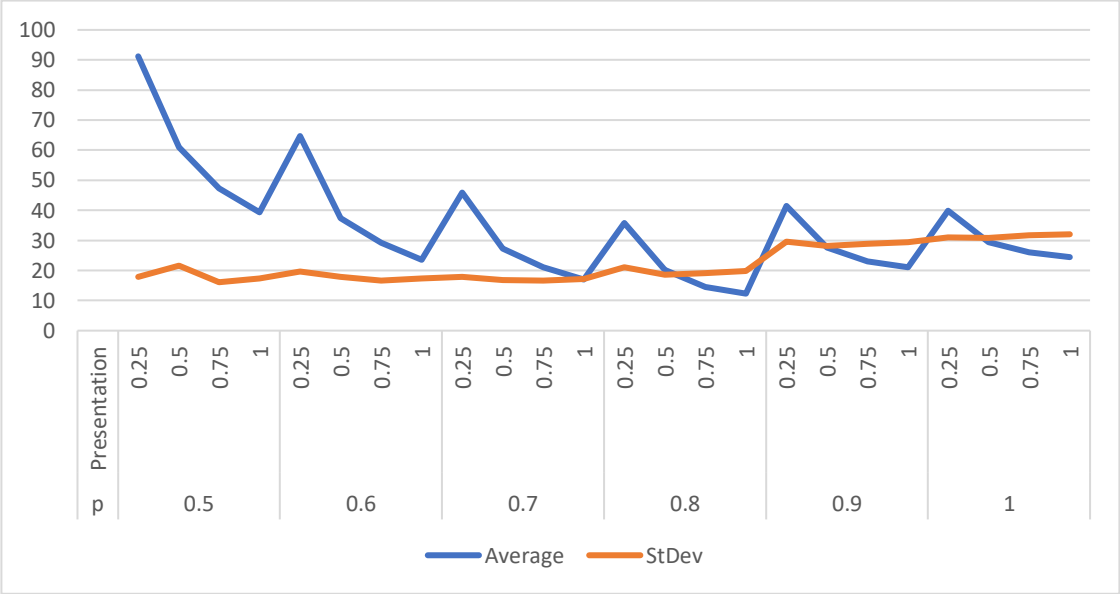


Figure 7. *Average and Standard Deviation for Parametric Hyperbolic LRS training*

When comparing the results of the optimized exponent parameter *p* on the test images to the results of Thompson, Celebi, and Buck (2020), there are many cases where this LRS does better. In many cases, this LRS gives MSE values that are more competitive to the batch k-means algorithm than then original hyperbolic schedule used by Thompson, Celebi, and Buck. However, the drawback of this parametric hyperbolic LRS is that it will not outperform the original hyperbolic with respect to computational time. This is because, the parametric hyperbolic LRS utilizes two parameters (exponent and number of clusters), while the standard hyperbolic LRS uses only one parameter

(exponent). The additional parameter in the parametric LRS, however, translates to only one extra addition operation. Therefore, as Table 7 shows, the CPU time difference between the resulting CQ methods is, in fact, negligible.

Alternative Parametric Hyperbolic

This modified version of the hyperbolic LRS behaved similarly to the original hyperbolic unlike the previous modified hyperbolic LRS. With this LRS, the exponent *p* = 0.5 rate gave the lowest average and standard deviation MSE values mirroring the results given by hyperbolic LRS. Also, like hyperbolic, the average and standard deviation MSE values for this alternative method increased as the exponent parameter *p* increased. This LRS is the only schedule where the best MSE was achieved using only half of the image (that is, presentation factor = 0.5). The parametric constant *pc* was only tested with the values 1, 2, and 3, but the data shows that as the value increases, typically the MSE decreases. And lastly, as the parametric fraction *pf* approached 1, the MSE decreased as well, see Figure 8.

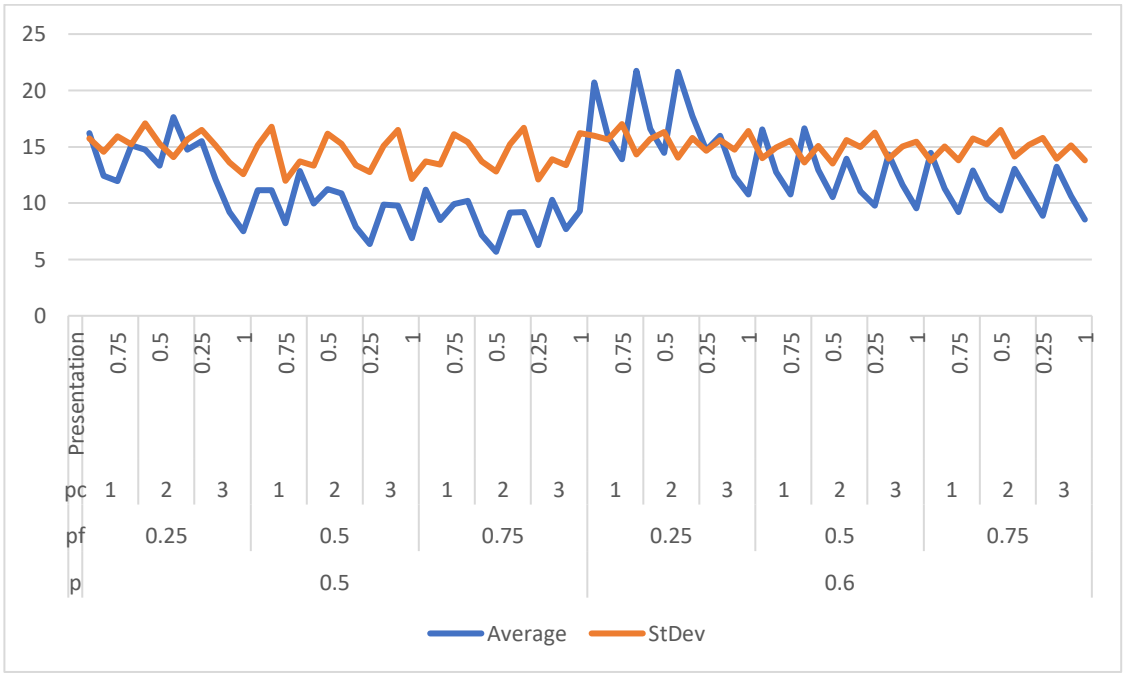


Figure 8. *Average and Standard Deviation for Alternative Parametric Hyperbolic LRS training*

When using the optimized parameters on the 8 test images, this was the only LRS that was able to achieve respectable results by using only 50% of the image rather than the full image. Because it only utilized half of the image, this LRS showed the fastest execution time for the three images (Baboon, Lenna, and Peppers) despite being the most complex LRS. Compared to other LRS's, the alternative parametric hyperbolic LRS is about 10% faster, whereas compared to BKM (batch k-means), which is the gold standard algorithm in CQ, it is between 19 and 44 times faster.

CHAPTER 5 CONCLUSIONS AND FUTURE WORK

In this thesis, alternative LRS's were explored for use in a CQ method based on MacQueen's k-means clustering algorithm. First, the maximin initialization method was used as a means to select an effective initial color palette. Then, the image was subsampled *quasirandomly* using a Sobol' sequence to present the pixels to the clustering algorithm in a deterministic fashion. Finally, the MacQueen's k-means clustering algorithm was implemented with various LRS's with differing complexities to determine the effects and viability of each LRS. A comprehensive experiment was conducted on 20 training and 8 test images that are commonly used in the CQ literature. The results showed that, with respect to quantization quality, several of the investigated LRS's are comparable to the hyperbolic LRS proposed by Thompson, Celebi, and Buck's (2020). In addition, one of the investigated LRS's outperformed the hyperbolic LRS in terms of CPU time.

Further work includes investigation of alternative, faster initialization methods as this thesis focused solely on the maximin initialization algorithm. Celebi, Kingravi, and Vela (2013) describes numerous other initialization methods that could be profitably applied to the presented CQ method. Finally, more advanced online unsupervised learning methods could be explored as discussed in Yair, Zeger, and Gersho (1992).

REFERENCES

Bratley, P., Fox, B. L. (1988). Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. *ACM transactions on Mathematical Software, 14*(1) 88–100. doi: 10.1145/42288.214372

Braudaway, G. W. (1987). Procedure for Optimum Choice of a Small Number of Colors from a Large Color Palette for Color Imaging. *Proceedings of the Electronic Imaging Conference*, 71–75

Brun, L., & Mokhtari, M. (2000). Two High Speed Color Quantization Algorithms. *Proceedings of the 1st International Conference on Color in Graphics and Image Processing*, 116–121

Brun, L., & Trémeau, A. (2003). Color Quantization. *Digital Color Imaging Handbook* (G. Sharma, Ed.), CRC Press, 589–638

Celebi, M. E. (2009). Fast Color Quantization Using Weighted Sort-Means Clustering. *Journal of the Optical Society of America A, 26*(11), 2434–2443. doi: 10.1364/JOSAA.26.002434

Celebi, M. E. (2011). Improving the Performance of K-Means for Color Quantization. *Image and Vision Computing, 29*(1), 260–271. doi: 10.1016/j.imavis.2010.10.002

Celebi, M. E., Hwang, S., & Wen, Q. (2014). Color Quantization Using the Adaptive Distributing Units Algorithm. *Imaging Science Journal, 62*(2), 80–91. doi: 10.1179/1743131X13Y.0000000059

Celebi, M. E., Kingravi, H., & Vela, P. A. (2013). A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm. *Expert Systems with Applications, 40*(1), 200–210. doi: 10.1016/j.eswa.2012.07.021

Celebi, M. E.(ed.) (2015) Partitional Clustering Algorithms. Springer. doi: 10.1007/978-3-319-09259-1

Darken, C., Moody, J. (1990) Fast adaptive K-means clustering: some empirical results. *Proceedings of the 1990 International Joint Conference on Neural Networks*. 233–238 doi: 10.1109/IJCNN.1990.137720

Dekker, A. (1994). Kohonen Neural Networks for Optimal Colour Quantization. *Network: Computation in Neural Systems, 5*(3), 351–367. doi: 10.1088/0954-898X/5/3/003

El-Said, S. A. (2015). Image Quantization Using Improved Artificial Fish Swarm Algorithm. *Soft Computing, 19*(9), 2667–2679. doi: 10.1007/s00500-014-1436-0

Fritzke, B. (1997). Some Competitive Learning Methods. Technical Report, *Institute for Neural Computation, Ruhr-Universität Bochum*

Gervautz, M., & Purgathofer, W. (1988). A Simple Method for Color Quantization: Octree Quantization. In N. Magnenat-Thalmann & D. Thalmann (Eds.), *New Trends in Computer Graphics* (pp. 219–231). Berlin, Germany: Springer. doi: 10.1007/978-3-642-83492-9_20

Gonzalez, T. (1985). Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science 38*(2–3), 293–306. doi: 10.1016/0304-3975(85)90224-5

Heckbert, P. (1982). Color Image Quantization for Frame Buffer Display. *Proceedings of ACM SIGGRAPH Computer Graphics, 16*(3), 297–307. doi: 10.1145/965145.801294

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). ACM Computing Surveys (CSUR). *Data Clustering: A Review 31*(3), 264–323. doi: 10.1145/331499.331504

Joy, G., & Xiang, Z. (1993). Center-Cut for Color Image Quantization. *Visual Computing, 10*(1), 62–66. doi: 10.1007/BF01905532

Khaled, A., Abdel-Kader, R. F., & Yasein, M. S. (2016). A Hybrid Color Image Quantization Algorithm Based on k-Means and Harmony Search Algorithms. *Applied Artificial Intelligence, 30*(4), 331–351. doi: 10.1080/08839514.2016.1169049

Linde Y., Buzo A., & Gray, R. (1980) An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications, 28*(1), 84–95, doi: 10.1109/TCOM.1980.1094577

Lloyd, S. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory, 28*(2), 129–136. doi: 10.1109/TIT.1982.1056489

MacQueen, J. (1967) Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* 281–297 doi: 10.1.1.308.8619

Ozturk, C., Hancer, E., & Karaboga, D. (2014). Color Image Quantization: A Short Review and an Application with Artificial Bee Colony Algorithm. *Informatica, 25*(3), 485–503. doi: 10.15388/Informatica.2014.25

Pérez-Delgado, M. L. & Gallego J. A. R. (2018). A two-stage method to improve the quality of quantized images. *Journal of Real-Time Image Processing*. doi: 10.1007/s11554-018-0814-8

Pérez-Delgado, M. L. & Gallego, J. A. R. (2019). A Hybrid Color Quantization Algorithm That Combines the Greedy Orthogonal Bi-Partitioning Method With Artificial Ants. *IEEE Access, 7*, 128714– 128734. doi: 10.1109/ACCESS.2019.2937934

Pérez-Delgado, M. L. (2015). Color Image Quantization Using the Shuffled-Frog Leaping Algorithm. *Engineering Applications of Artificial Intelligence, 79*, 142–158. doi: 10.1016/j.engappai.2019.01.002

Pérez-Delgado, M. L. (2015). Colour Quantization with Ant-tree. *Applied Soft Computing, 36*, 656–669. doi: 10.1016/j.asoc.2015.07.048

Pérez-Delgado, M. L. (2018). Artificial ants and fireflies can perform colour quantisation. *Applied Soft Computing, 73*, 153–177. doi: 10.1016/j.asoc.2018.08.018

Pérez-Delgado, M. L. (2019). The color quantization problem solved by swarm-based operations. *Applied Intelligence, 49*, 2482–2514. doi: 10.1007/s10489-018-1389-6

Pérez-Delgado, M. L. (2020). Color Quantization with Particle Swarm Optimization and Artificial Ants. *Soft Computing, 24*, 4545–4573. doi: 10.1007/s00500-019-04216-8

Robbins, H. & Monro, S. (1951) A Stochastic Approximation Method. *Annals of Mathematical Statistics, 22*(3), 400–407. doi:10.1214/aoms/1177729586

Su, Q., & Hu, Z. (2013). Color Image Quantization Algorithm Based on Self-Adaptive Differential Evolution. *Computational Intelligence and Neuroscience, 2013,* Article ID 231916, 8 pages. doi: 10.1155/2013/231916

Thompson, S., Celebi, M. E. & Buck, K. H. (2020) Fast color quantization using MacQueen's k-means algorithm. *Journal of Real-Time Image Processing* doi:10.1007/s11554-019-00914-6

Uchiyama, T., & Arbib, M. (1994). An Algorithm for Competitive Learning in Clustering Problems. *Pattern Recognition*, *27*(10), 1415–1421, doi: 10.1016/0031-3203(94)90074-4

Wan, S. J., Prusinkiewicz, P., & Wong, S. K. M. (1990). Variance-Based Color Image Quantization for Frame Buffer Display. *Color Research and Application*, *15*(1), 52–58. doi: 10.1002/col.5080150109

Wen, Q., & Celebi, M. E. (2011). Hard versus Fuzzy C-Means Clustering for Color Quantization. *EURASIP Journal on Advances in Signal Processing*, 2011, 118–129. doi: 10.1186/1687-6180-2011-118

Wu, K. L., Yang, M. S. (2006). Alternative learning vector quantization. *Pattern Recognition, 39*(3), 351–362 doi: 10.1016/j.patcog.2005.09.011

Wu, X., Kumar, V., Quinlan, J. R., *et al*. (2008). Top 10 Algorithms in Data Mining. *Knowledge and Information Systems, 14,* 1–37. doi: 10.1007/s10115-007-0114-2

Xiang, Z. (1997). Color Image Quantization by Minimizing the Maximum Intercluster Distance. *ACM Transactions on Graphics*, *16*(3), 260–276. doi: 0.1145/256157.256159

Yair, E., Zeger, K., & Gersho, A. (1992). Competitive Learning and Soft Competition for Vector Quantizer Design. *IEEE Transactions on Signal Processing, 40*(2), 294–309 doi: 10.1109/78.124940

Yang, C. Y., & Lin, J. C. (1996). RWM-Cut for Color Image Quantization. *Computers & Graphics*, *20*(4), 577–588. doi: 10.1109/ICDAR.1995.601984