EFFECTIVE COLOR QUANTIZATION USING

CORESET SAMPLING

by
German Valenzuela

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

Master of Science
in
Applied Computing

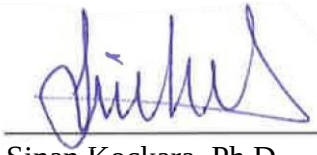
Conway, Arkansas
December 2017

TO THE OFFICE OF GRADUATE STUDIES:

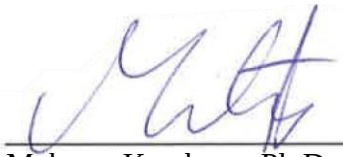
The members of the Committee approve the thesis of

German Valenzuela presented on November 30, 2017.

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



Sinan Kockara, Ph.D.



Mahmut Karakaya, Ph.D.

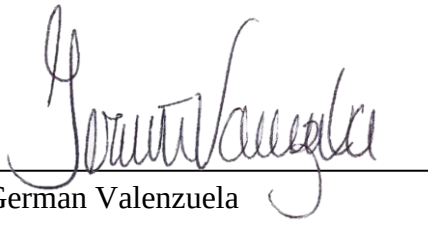
PERMISSION

Title Effective Color Quantization Using Coreset Sampling

Department Computer Science

Degree Master of Science

In presenting this thesis/dissertation in partial fulfillment of the requirements for 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.



German Valenzuela

November 30, 2017

ACKNOWLEDGMENTS

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. The door to Prof. Celebi's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed the project and this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge Dr. Sinan Kockara of the Department of Computer Science at University of Central Arkansas as the second reader of this thesis. I am gratefully indebted to him for his very valuable comments on this thesis and for his support throughout my graduate studies.

Finally, I must express my very profound gratitude to my family and to my spouse for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

German Valenzuela

VITA

German Valenzuela was born in Cd. Juarez, Mexico. He attended elementary and middle school in Mexico before immigrating to the United States. He completed his GED in 2012, with a high score that awarded him the GED Achievement Scholarship at the University of Arkansas Community College at Morrilton. He attended UACCM for a year before transferring to the University of Central Arkansas. After the transfer, he graduated from UCA with a Bachelor of Science in Computer Science with a minor in Mathematics in April 2016.

ABSTRACT

Color quantization is a technique to reduce the number of colors in a digital color image. Although the hardware constrains that ensued the need for color quantization are uncommon nowadays, color quantization remains an important image processing technique. In this thesis, a novel, fast, and effective color quantization method based on the k-means algorithm is introduced. The proposed method utilizes careful initialization, data subsampling, and coreset construction to attain high quality and high speed quantization. Tests on various well-known, publicly available images demonstrate that the proposed method outperforms k-means in terms of speed while delivering nearly identical results.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A true color image is generally comprised of pixels with three components: red, green, and blue, each of which is typically represented by 1 byte. Thus, 3 bytes are required to store each color pixel. This yields 2^{24} (approximately 16.8 million) possible color combinations. Color quantization is an image processing technique that reduces the number of unique colors in a digital color image, thereby allowing true color images to be stored and displayed using only a small number of colors. Color quantization was originally used to satisfy display hardware constraints. Although 24-bit true-color displays are common today, color quantization is still an important step in image processing and computer graphics. Some of its uses include compression, segmentation, text localization/detection, color texture analysis, watermarking, non-photorealistic rendering, and content-based retrieval (Celebi, 2011).

Quantization Methods

Color quantization methods can be broadly classified in two categories according to their initial palette selection scheme: image-independent and image-dependent. Image-independent methods (Gentile, Allebach, & Walowit, 1990), such as uniform quantization, achieve color quantization regardless of the original image's color distribution. These methods produce lower quality results. Nevertheless, they provide results in real-time. Image-dependent methods obtain the reduced color palette by analyzing the image itself. These methods produce high-quality results at the expense of higher computational requirements (Berge & Berger, 2009). Image-dependent methods can be categorized into preclustering (or hierarchical) and postclustering (or partitional)

approaches (Celebi, 2011). Preclustering methods are typically based on the statistical analysis of the original image's color distribution. Divisive preclustering procedures start with one cluster that contains all *n* image pixels. This cluster is then repeatedly divided until *k* clusters are obtained, where *k* is the target number of colors. In contrast, agglomerative preclustering methods start with *n* clusters, each containing one pixel. These are then combined repeatedly until *k* clusters remain. Postclustering algorithms find all *k* clusters simultaneously as a partition of the data, without imposing a hierarchical structure. These algorithms, although time consuming, produce better results than preclustering approaches as they start with an initial color palette and then iteratively improve it. In this thesis, we focus on postclustering methods.

Data Clustering as Means of Color Quantization

Since the color of a pixel is comprised of three components, red, green, and blue, color quantization can be viewed as a clustering problem in three dimensions (Celebi, 2009). The task is to identify the clusters that are most representative of the colors in an image. The color quantization process consists of two main steps that correspond to the steps in partitional clustering. The first step is to choose a color palette (a.k.a. initialization or seeding), with a smaller number of colors than that of the original image – typically in the range of 8–256. The second step is to map each original image pixel to one of the colors in the reduced palette which is achieved by clustering the pixel data.

CHAPTER 2

RELATED WORK

Color Quantization

Numerous color quantization techniques exist in the literature, such as popularity (Heckbert, 1982), median-cut (Heckbert, 1982), modified popularity (Braudaway, 1987), octree (Gervautz & Purgathofer, 1988), variance-based method (Wan, Prusinkiewicz, & Wong, 1990), greedy orthogonal bipartitioning (Wu, 1991), center-cut (Joy & Xiang, 1993), self-organizing map (Dekker, 1994), radius-weighted mean-cut (Yang & Lin, 1996), modified maximin (Xiang, 1997), pairwise clustering (Velho, Gomez, & Sobreiro, 1997), split and merge (Brun, & Mokhtari, 2000), Cheng and Yang (Cheng & Yang, 2001), weighted sort-means (Celebi, 2009), modified weighted sort-means (Celebi, 2011), fuzzy c-means (Wen & Celebi, 2011), adaptive distributing units (Celebi, Hwang, & Wen, 2014), and variance-cut (Celebi, Wen, & Hwang, 2015).

Heckbert (1982) proposed two color quantization methods – popularity and median-cut. Popularity builds a $16 \times 16 \times 16$ color histogram using 4 bits/channel uniform quantization and then takes the k most frequent colors in the histogram as the color palette. Median-cut starts by building a $32 \times 32 \times 32$ color histogram using uniform quantization. This histogram volume is then recursively split into smaller boxes until k boxes are obtained. At each step, the box that contains the greatest number of colors is split along the longest axis at the median point, so that the resulting sub-boxes each contain approximately the same number of colors. The centroids of the final k boxes are taken as the color palette.

Braudaway (1987) introduced the modified popularity method. This method starts

by building a $2^R \times 2^R \times 2^R$ color histogram using R bits per channel uniform quantization. It chooses the most frequent color as the first palette color c_1 and then reduces the frequency of each color c by a factor of $\left(1 - e^{\alpha\|c-c1\|^2}\right)$, where α is a user-defined parameter. The remaining palette colors are chosen similarly.

Gervautz and Purgathofer (1988) proposed the octree method. This two-phase method first builds an octree—a tree data structure in which each internal node has up to eight children—that represents the color distribution of the input image. Then, starting from the bottom of the tree, it merges the adjacent colors with the least number of pixels to the closest cluster until *k* colors are obtained.

Wan, Prusinkiewicz, and Wong (1990) introduced the variance-based method. This method is very similar to median-cut. At each step of this method, the box with the greatest error is split along the axis with the least weighted sum of projected variances at the point that minimizes the marginal error.

Wu (1991) proposed the greedy orthogonal bipartitioning procedure. It is analogous to the variance-based method, except at each step, the box is split along the axis that minimizes the sum of variances on both sides.

Joy and Xian (1993) presented the center-cut method. It is comparable to the median-cut method, except at each step, the box with the greatest range on any coordinate axis is split along its longest axis at the mean point cut.

Dekker (1994) proposed the self-organizing map scheme. It utilizes a one-dimensional self-organizing map with *k* neurons. A random subset of *n / f* pixels (*f* is a sampling factor greater than or equal to 1) is used in the training phase and the final weights of the neurons are taken as the color palette.

Yang and Lin (1996) proposed the radius-weighted mean-cut method. This method is nearly equivalent to the variance based method. The exception is that the box is split along the vector from the origin to the radius-weighted mean (RWM) at the RWM point.

Xiang (1997) introduced the modified maximin method. This method chooses the first palette color c_1 arbitrarily from the input image colors. The i-th color c_i $(i = 2, 3, \ldots, k)$ is then chosen to be the color that has the greatest minimum weighted Euclidean distance to the previously selected colors. The weights for the red, green, and blue channels are taken as 0.5, 1.0, and 0.25 respectively. Each of these initial palette colors is then recalculated as the mean of the colors assigned to it.

Velho, Gomez, and Sobreiro (1997) introduced pairwise clustering as an adaptation of Ward's agglomerative hierarchical clustering method (1963) to color quantization. It builds a $2^R \times 2^R \times 2^R$ color histogram and constructs a $Q \times Q$ joint quantization error matrix where Q is the number of colors in the reduced color histogram. The clustering procedure starts with Q singleton clusters, each of which contains one image color. In each iteration, the pair of clusters with the least joint quantization error is merged. This merging process is repeated until k clusters remain.

Brun and Mokhtari (2000) devised the split and merge method. This two-phase method first partitions the color space uniformly into B partitions. This initial set of B clusters is represented as an adjacency graph. Then, $(B - k)$ merge operations are performed to obtain the final k clusters. At each step of the second phase, the pair of clusters with the least joint quantization error is merged.

Cheng and Yang (2001) developed their self-named method. This method is

equivalent to the variance based-method, with the exception that at each step the box is split along a specially chosen line defined by the mean color and the color that is farthest from it at the mean point.

Celebi (2009, 2011) introduced the weighted sort-means procedure. This method is an efficient adaptation of the conventional k-means clustering algorithm to color quantization. It involves data reduction, sample weighting, and accelerated nearest neighbor search.

Wen and Celebi (2011) adapted the fuzzy c-means clustering algorithm (Bezdek, 1981) to color quantization. This algorithm is a modification of the (hard) c-means (or k-means) algorithm in which points can belong to more than one cluster. Its goal is to create optimal fuzzy C-partitions of the data set by minimizing the following objective function $J_m(\boldsymbol{U},\boldsymbol{V})\sum_{k=1}^{N}\sum_{i=1}^{C}(u_{ik})^m(d_{ik})^2$, where N is the number of pixels, C is the target number of clusters/colors, $1 \leq m < \infty$ controls the degree of fuzziness, $\boldsymbol{U}$ is the fuzzy partition matrix, and $\boldsymbol{V}$ is the prototype matrix.

Celebi, Hwang, and Wen (2014) proposed the adaptive distributing units (ADU) color quantization method as an adaptation of Uchiyama and Arbib's clustering algorithm (1994). ADU is a competitive learning algorithm in which units compete to represent the input point presented in each iteration. The winner is then rewarded by moving it closer to the input point at a rate of γ (the learning rate). The procedure starts with a single unit whose center is given by the centroid of the input points. New units are added by splitting existing units that reach a user-defined number of wins until the number of units reaches K.

Finally, Celebi, Wen, and Hwang (2015) proposed an effective color quantization

method based on divisive clustering named the variance-cut. This algorithm starts by building a $32 \times 32 \times 32$ color histogram using 5 bits per channel uniform quantization. In each iteration, the partition with the greatest error is split along the coordinate axis with the greatest variance at the mean point. After $k - 1$ splits, the centroids of the resulting k subpartitions are taken as the color palette.

Many of the most recent color quantization methods are based on metaheuristics (Blum & Roli, 2003) or a hybrid of a classical clustering algorithm (e.g., k-means and fuzzy c-means) and metaheuristics. These methods formulate color quantization as a global optimization problem and then attempt to solve it using a variety of nature-inspired metaheuristics (simulated annealing, genetic algorithms, differential evolution, etc.). These algorithms can be fairly effective, but this comes at the expense of a significant computational burden (e.g., they can be orders of magnitude slower than k-means). In the following, we briefly mention a few representative color quantization methods based on metaheuristics.

Su and Hu (2013) proposed a hybrid method that combines k-means clustering algorithm and self-adaptive hybrid differential evolution. Ozturk *et al.* (2014) proposed a method based on the artificial bee colony algorithm. Schaefer and Nolle (2015) proposed a method that minimizes the S-CIELAB image quality metric using the step width adapting simulated annealing algorithm. Pérez-Delgado (2015) proposed a method based on the ant-tree algorithm. El-Said (2015) proposed a hybrid method that combines the fuzzy c-means clustering algorithm and the artificial fish swarm algorithm. Khaled *et al.* (2016) proposed a hybrid method that combines k-means clustering algorithm and harmony search algorithm. Finally, Hu *et al.* (2016) presented a method that minimizes

intracluster distance while maximizing intercluster separation. They tackled this multiobjective optimization problem using a self-adaptive hybrid differential evolution approach.

Coresets

Coresets are concise summaries of large data sets. Coreset construction and its applications have been considered in many contexts including k-means, principal component analysis and projective clustering (Feldman, Schmidt, & Sohler, 2013), real-time data segmentation and summarization (Volkov, 2016), vector summarization applied to network graphs (Feldman, Ozer, & Rus, 2017), and machine learning (Bachem *et al.*, 2017).

Feldman *et al.* (2013) proposed coreset construction algorithms that produce coresets of constant size that is independent of the complete data set size n, and dimensionality d. Particularly beneficial when $d \sim n$, these coresets allow the handling of k-means, principal component analysis, and projective clustering with update time and memory that is polynomial in $log\ n$ and only linear in d.

Volkov (2016) developed a group of real-time data reduction algorithms for big data streams through coreset construction. These coresets were then used to perform effective analysis such as segmentation, summarization, classification, and prediction. Volkov proposed a theoretical framework for the various coreset construction algorithms that can handle boundless, real-time data streams, and is easily scalable and parallelizable.

Feldman *et al.* (2017) presented coresets as a deterministic data summarization algorithm that aims to approximate the mean $\bar{p}$ of a complete set, by a weighted mean $\tilde{p}$

that is independent of both the size of the complete set and its dimensionality. The main application of this algorithm is to build a sparse social graph from the GPS location data of smart-phone users. This graph is then used to recognize and forecast various activities such as meetings, friend groups, and gathering places.

Bachem *et al.* (2017) introduced a coreset construction algorithm based on importance-weighted subsampling that is applicable to a variety of machine learning problems such as maximum likelihood estimation of mixture models, Bayesian non-parametric models, principal component analysis, regression, and general empirical risk minimization.

The work presented in the next chapter builds on previous research and develops a novel, fast, and effective color quantization method based on a recently proposed coreset construction method. We propose an approach similar to weighted sort-means (Celebi, 2011) in that it combines various performance enhancing techniques: data reduction, effective initialization, and an efficient k-means clustering algorithm. However, unlike the weighed sort-means method, a coreset will be constructed and the data clustering algorithm will be executed on this reduced data set.

CHAPTER 3

PROPOSED COLOR QUANTIZATION METHOD

In this chapter, the implementation of the proposed color quantization method is explained. First, a deterministic decimation method that reduces the size of the input image is described. Then, a fast and effective cluster center initialization method is introduced. Thereafter, coresets are introduced and a coreset construction algorithm is presented as a means of further reducing computational time. Finally, the data clustering algorithm is described.

Decimation

One of the simplest ways to improve the speed of the k-means algorithm is to reduce the amount of data to be processed. Let f be the decimation factor desired. A deterministic decimation method can be implemented in which only rows and columns that are multiples of f, that is, rows/columns 0, f, 2f, …, will be sampled to form the initial data set for clustering. Since color images contain many redundant colors especially within small neighborhoods, this kind of subsampling is very effective in reducing the computational time without degrading quantization quality in an appreciable manner.

Initialization

A common approach to determine the initial cluster centers is to select k points uniformly at random from the complete data set and take these as the initial cluster centers (Celebi *et al.*, 2013).

According to Celebi *et al.* (2013), k-means is relatively sensitive to initialization. Some of the negative effects of improper initialization include empty clusters, slower convergence, and a higher probability of getting stuck at a bad local minima. Arthur and

Vassilvitskii (2007) proposed an adaptive initialization method to address these drawbacks and improve the overall performance of k-means. This method, simply referred to as k-means++, is described in Table 1.

Table 1

The k-means++ algorithm

Step	Description
1a	Take one center c_1, chosen uniformly at random from X.
1b	Take a new center c_i, choosing $x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$
1c	Repeat Step 1b. until we have taken k centers altogether.

In this algorithm, the first center is chosen uniformly at random from the data set. The remaining $k-1$ centers are then chosen with a probability proportional to the squared Euclidean distance from the centers already chosen. This weighting is referred to as simply "D^2 weighing". This weighted sampling ensures that points that are well separated are more likely to be selected as initial center centers.

Coreset Construction

Originally studied in computational geometry, coresets relied on computationally expensive methods (Har-Peled, 2011). It is just recently that coresets have evolved to utilize a sampling-based approach that allows practical construction for various applications.

Although aimed toward machine learning problems, the theoretically-comprehensive framework for coreset construction introduced by Bachem *et al.* (2017) is applicable to general problem-solving. Given a data set and a particular problem, a

coreset of the former for the latter gives a solution that is provably competitive with the solution found on the former. The coreset construction procedure, detailed in Table 2, consists of two key steps. First, the importance of the different data points with regards to the objective function and optimal solution is determined. Then, this importance information is used to select a coreset by means of importance sampling.

Table 2

Coreset construction algorithm

Step	Description
Require X, k, B, m	
1	$\alpha \leftarrow 16(\log k + 2)$
2	For each b_i in B do
3	$B_i \leftarrow$ Set of points from X closest to b_i in terms of d. Ties broken arbitrarily.
4	$c_\phi \leftarrow \frac{1}{\lvert X \rvert}\sum_{x' \in X} d(x', B)$
5	For each $b_i \in B$ and $x \in B_i$ do
6	$s(x) \leftarrow \frac{\alpha\, d(x,B)}{c_\phi} + \frac{2\alpha \sum_{x' \in B_i} d(x', B)}{\lvert B_i \rvert c_\phi} + \frac{4\lvert X \rvert}{\lvert B_i \rvert}$
7	For each $x \in X$ do
8	$p(x) \leftarrow s(x) / \sum_{x' \in X} s(x')$
9	$C \leftarrow$Sample m weighted points from X where each point x has weight $\frac{1}{m \cdot p(x)}$ and is sampled with probability $p(x)$
10	Return C

The algorithm first selects a set of k centers through D^2 sampling, as detailed in Table 1. This set is then used to compute the sensitivity $s(x)$ of each point *x*. Sensitivity is the worst-case impact of each data point on the objective function. Finally, m points are sampled where each point is included in the coreset with probability proportional to the normalized sensitivity $p(x)$. In this coreset, point *x* is assigned a weight of $(m \cdot p(x))^{-1}$ where m is the size of the coreset.

The research below proposes the use of the aforementioned algorithm to construct such a summary data set and solve the color quantization problem on this smaller data set. In this implementation, the coreset size m is defined as $m = n \cdot cf$; where n is the size of original data set and cf, or coreset fraction, is a number that satisfies $0 < cf \leq 1$.

Clustering

Perhaps, the most frequently used clustering method in color quantization is the Lloyd's algorithm (1982). Commonly referred to as k-means, Lloyd's method starts with a given integer value *k*, which denotes the number of clusters or colors in this context. The algorithm then assigns each data point *x*, in the data set, to the closest cluster by minimizing the Sum of Squared Error (SSE) defined as:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} d\left(\boldsymbol{x}, \overline{\boldsymbol{x}_{C_i}}\right)^2,$$

where $\overline{\boldsymbol{x}_{C_i}}$ denotes the centroid (or center of mass) of cluster C_i. To determine the closest cluster, the Euclidean distance (*d*) between the point and each cluster's centroid is computed. K-means is illustrated in Table 3.

Table 3

The k-means algorithm

Step	Description
1	Randomly choose an initial set of k centers $C = \{c_1, c_2, \cdots, c_k\}$
2	For each $i \in \{1, \ldots, k\}$, set the cluster C_i to be the set of points in X that are closer to c_i than they are to c_j for all $j \neq i$
3	For each $i \in \{1, \ldots, k\}$, set c_i to be the centroid of all points in C_i: $c_i = \frac{1}{\lvert C_i \rvert} \sum_{x \in C_i} x$
4	Repeat Steps 2 and 3 until a user-defined termination criterion is satisfied

The algorithm first selects a set of *k* centers uniformly at random, unless a

different initialization scheme is implemented. Then the squared Euclidean distance between each point and each center is computed. Each point is assigned to the cluster to which it is closest. At the end of each iteration, each new cluster centroid is computed to be the centroid of all the points that belong to the cluster. This algorithm continues until the relative improvement in the SSE drops below a user-defined threshold *T*, that is, $(SSE_{i-1} - SSE_i)/SSE_i \leq T$, where $0 < T < 1$.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

This chapter describes the experimental procedure followed and then presents a discussion of the results. First, the input data sets for both the proposed and the comparison algorithms are outlined and their use is explained. Then, a brief description of the performance measures collected, and their significance, is presented. Thereafter, the variables as well as the constant parameters utilized in the experiments are outlined. Finally, the results of the experiment are discussed and the data gathered during the experiment is presented in the form of tables and figures.

Experimental Setup

First, the program reads the original image data from a PPM file. PPM images are amongst the simplest image file formats. The headers for such images contain a number that identifies the image format, the image width and height, and the maximum brightness value which determines the required number of bits-per-pixel for each channel. The proposed algorithm was tested on 8 commonly used, publicly available PPM images given in Table 4. The original image data set is then decimated, that is subsampled as described in Chapter 2, so as to reduce the amount of data to be processed. Then, the k-means++ algorithm is executed to select the initial set of centers. This decimated set and initial center set are the basis for all the subsequent processing for both k-means and the proposed algorithm. Hereinafter, the k-means method executed on the full set will simply be referred to as *k-means*, whereas the k-means method run on the coreset will be referred to as *coremeans*. The initial set of centers selected by k-means++ are then duplicated so that it can be used to initialize the coreset construction algorithm,

k-means, and coremeans. The coreset algorithm takes the decimated data set and the initial set of centers and constructs the coreset. Once the coreset is constructed, k-means is executed separately on the coreset and the full set. The input image is then quantized by a mapping function that replaces the original colors with the colors represented by the final centroids given by each execution. Next, the quantized images are compared to the original image by means of the Mean Absolute Error and Mean Squared Error measures described in the next section for each method (k-means and coremeans). The two quantized images are then written to separate PPM files. The previously mentioned steps are illustrated in Figure 1.

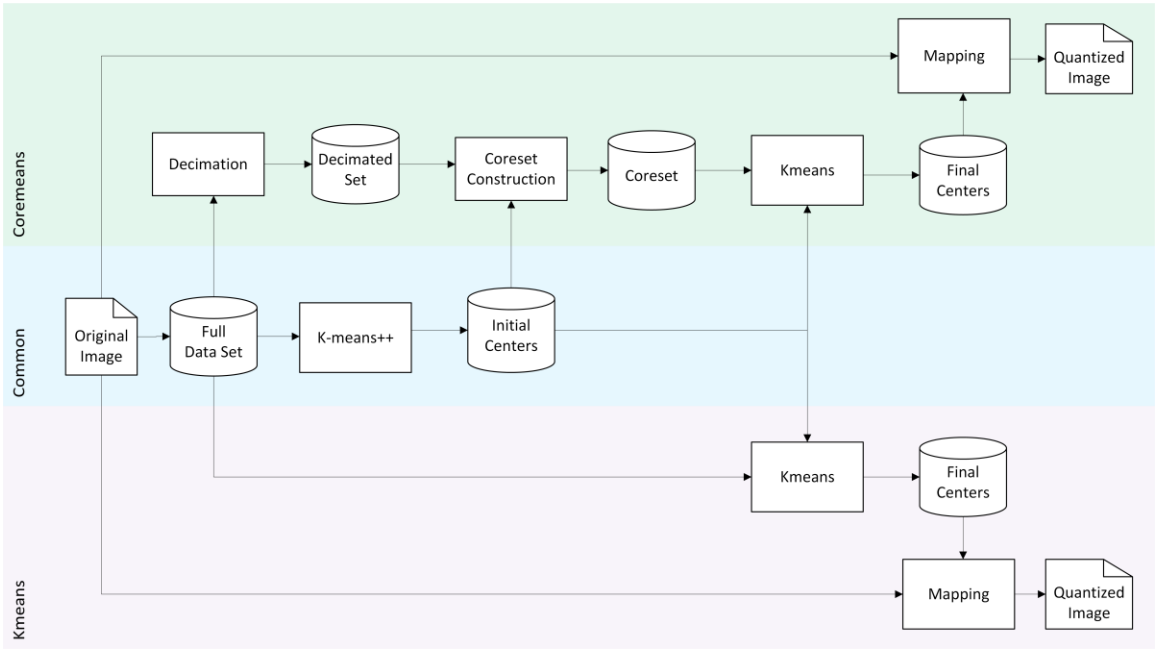
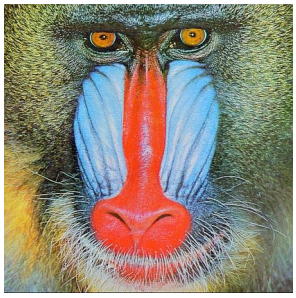
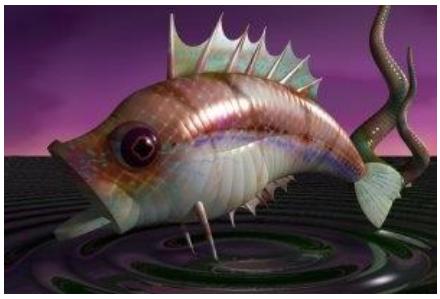


Figure 1. Experimental procedure flow chart.

Table 4

Image set

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

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

Performance Measures

Mean Absolute Error

The Mean Absolute Error (MAE) is a dimensioned measure of average model performance error. This measure is said to be "dimensioned" in that it expresses average model-prediction error in the units of the variable of interest. MAE is computed as follows:

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

where X and $\hat{X}$ denote, respectively, the H × W original and quantized images in the RGB color space (Celebi, 2011).

Mean Squared Error

One of the simplest and most extensively used image quality metrics is the Mean Squared Error (MSE). It is computed by averaging the squared intensity differences of corresponding pixels from two images. This quality measure is appealing because it is simple to calculate, has clear physical meaning, and is mathematically convenient in the context of optimization (Wang, Bovik, Sherikh, & Simoncelli, 2004). MSE is computed as follows:

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

where X and $\hat{X}$ denote, respectively, the H × W original and quantized images in the RGB color space (Celebi, 2011).

Time

The computational efficiency of an algorithm was measured by the CPU time it

uses. The data type *struct timeval*, which is part of the GNU C Library, was used to represent the elapsed time between the start of the considered functions and their end. Time measurements were computed using the *getrusage* function, which is able to measure CPU time to an accuracy of a microsecond. Time measurements were converted to milliseconds for display purposes. For the full set, initial center selection through k-means++, and k-means running times were added. For the coreset, initial center selection through k-means++, coreset construction, and k-means running times were considered.

Experimental Parameters

The variable parameters for k-means and coremeans were the following: desired number of colors, $k = \{32, 64, 128, 256\}$; decimation factor, $df = \{1, 2, 4\}$; and coreset fraction, $cf = \{0.125, 0.250, 0.375, 0.500\}$. The constant parameters used throughout all the test runs were as follows: the maximum number of k-means iterations per run, $I = 50$; number of runs, $R = 10$; and threshold for k-means convergence, $T = 10^{-3}$.

Each time the program was executed, MAE, MSE, and total running time for each method were collected. This data was used to calculate the mean and standard deviation for the collected performance measures for each image. The program was implemented in the C++ programming language, compiled with the GNU g++ compiler version 5.4.0, and executed on a 3.40GHz Intel Core i7-6700 CPU.

Discussion

The statistics based on the experimental results are given in Tables 5-16. Tables 5-8 contain MAE, Tables 9-12 present MSE, and Tables 13-16 display CPU time, in milliseconds, for each k value tested. Only one result is provided for k-means as the variable parameters do not influence the results of this algorithm. For coremeans, the best

results are displayed in bold. Based on the collected statistics, the following observations can be made:

Coremeans performs significantly worse regarding CPU time in certain scenarios; namely, when the decimation factor and the coreset fraction are close to 1. In such cases, the extra steps performed to construct the coreset are detrimental as the resulting coreset is nearly as large as the full set, see Figure 2.

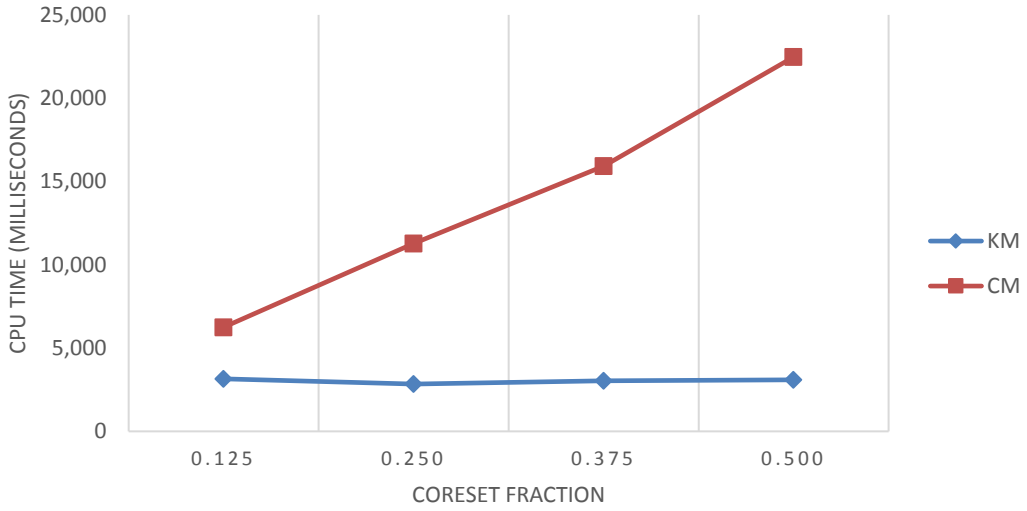


Figure 2. K-means vs. coremeans: CPU time for k = 256 and df = 1.

The decimation factor and coreset fraction parameters influence the quantization quality (as measured by MAE and MSE) in opposite ways in that the greater the former the lower the quality, whereas the greater the latter the higher the quality. For the proposed method to produce satisfactory results, the following must be true: $df \geq 2$ and $0 < cf \leq 0.5$. This effect is visualized in Figure 3.

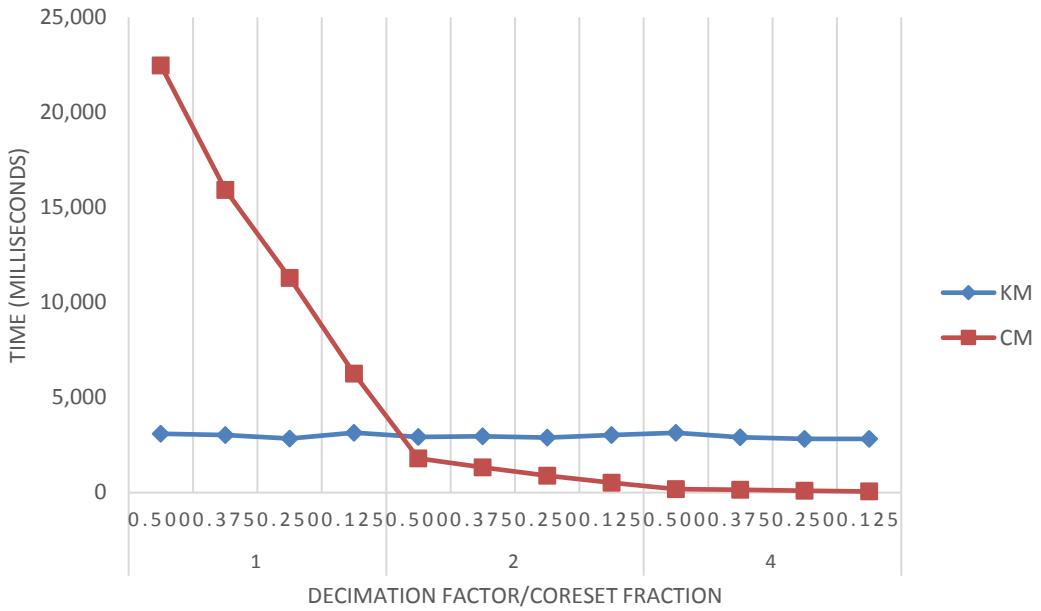


Figure 3. K-means vs coremeans: CPU time for various *cf* and *df* values

Clearly, the added burden of the coreset construction algorithm on CPU time can only be offset when the coreset fraction is small and decimation factor is large. In such scenarios, coremeans processing time diverges positively from k-means. In the experimental scenario illustrated in Figure 4, coremeans completed approximately 48 times faster than k-means.

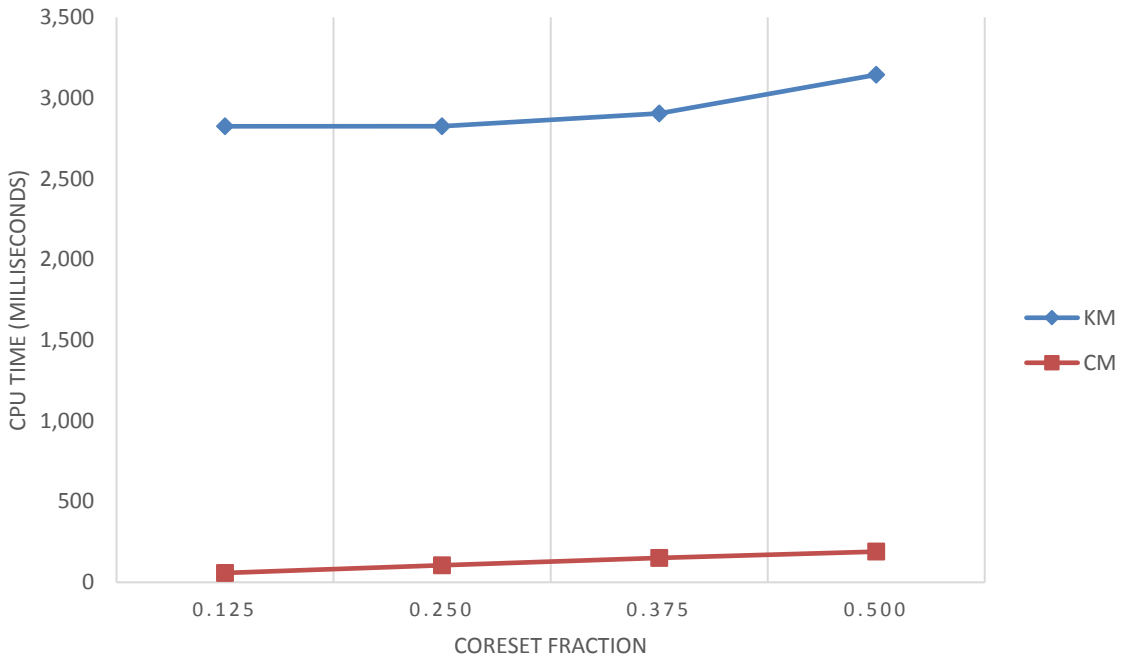


Figure 4. K-means vs coremeans: CPU time for $k = 256$ and $df = 4$.

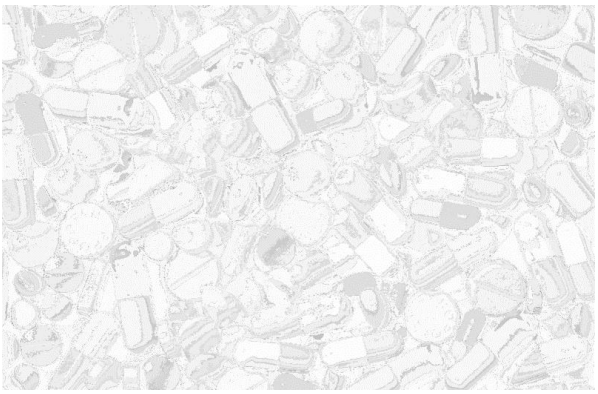
Based on the data collected, and with respect to the error measures used, the effectiveness of coremeans is very competitive with that of k-means. Although the coreset quantized image's MSE is oftentimes slightly higher than that of the k-means quantized image, visually, the differences between the results obtained are virtually indistinguishable.

Figures 5 to 8 illustrate some of the experimental results with various k values. It is evident that although there is a slightly higher MAE and MSE for the coremeans-quantized images, coremeans outperforms k-means with respect to CPU time in nearly all presented scenarios and this time difference grows considerably as k increases.

(a) Kmeans output image
Time = 647

(b) Kmeans error image
MAE = 19
MSE = 203

(c) Original image



(d) Kmeans vs coremeans error image

(e) Coremeans output image
Time = 68

(f) Coremeans error image
MAE = 20
MSE = 208

Figure 5. Pills input/output images, K = 32.

(a) Kmeans output image
Time = 1454

(b) Kmeans error image
MAE = 12
MSE = 85

(c) Original image

(d) Kmeans vs coremeans error image

(e) Coremeans output image
Time = 74

(f) Coremeans error image
MAE = 13
MSE = 91

Figure 6. Goldhill input/output images, K = 64.

(a) Kmeans output image
Time = 2285

(b) Kmeans error image
MAE = 10
MSE = 63

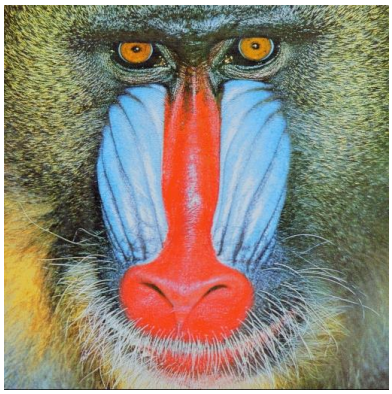
(c) Original image

(d) Kmeans vs coremeans error image

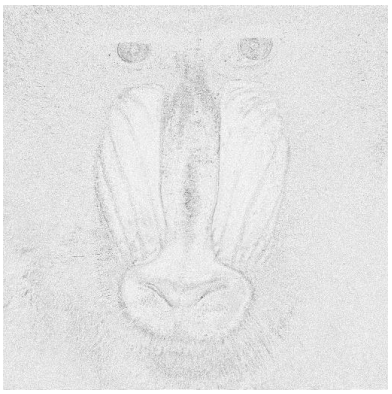
(e) Coremeans output image
Time = 72

(f) Coremeans error image
MAE = 11
MSE = 70

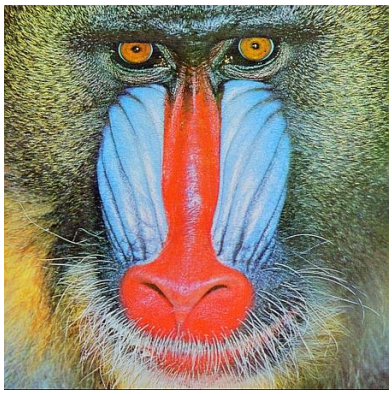
Figure 7. Motocross input/output images, K = 128.



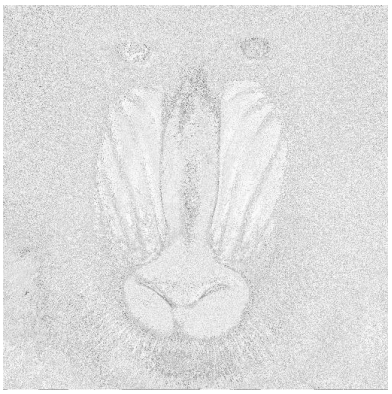
(a) Kmeans output image
Time = 3152



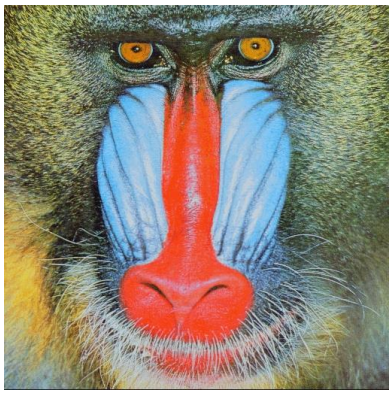
(b) Kmeans error image
MAE = 14
MSE = 97



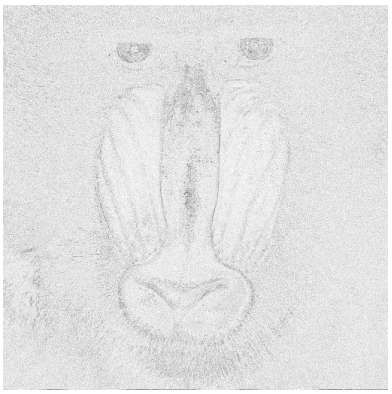
(c) Original image



(d) Kmeans vs coremeans error image



(e) Coremeans output image
Time = 58



(f) Coremeans error image
MAE = 15
MSE = 114

Figure 8. Baboon input/output images, K = 256.

Table 5

MAE, 32 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	27 ± 0	**27 ± 0**	Goldhill	1	.125	16 ± 0	**16 ± 0**
		.250		**27 ± 0**			.250		**16 ± 0**
		.375		**27 ± 0**			.375		**16 ± 0**
		.500		**27 ± 0**			.500		**16 ± 0**
	2	.125		**27 ± 0**		2	.125		**16 ± 0**
		.250		**27 ± 0**			.250		**16 ± 0**
		.375		**27 ± 0**			.375		**16 ± 0**
		.500		**27 ± 0**			.500		**16 ± 0**
	4	.125		**27 ± 0**		4	.125		**16 ± 0**
		.250		**27 ± 0**			.250		**16 ± 0**
		.375		**27 ± 0**			.375		**16 ± 0**
		.500		**27 ± 0**			.500		**16 ± 0**
Lenna	1	.125	15 ± 0	**15 ± 0**	Motocross	1	.125	18 ± 0	**18 ± 0**
		.250		**15 ± 0**			.250		**18 ± 0**
		.375		**15 ± 0**			.375		**18 ± 0**
		.500		**15 ± 0**			.500		**18 ± 0**
	2	.125		**15 ± 0**		2	.125		**18 ± 0**
		.250		**15 ± 0**			.250		**18 ± 0**
		.375		**15 ± 0**			.375		**18 ± 0**
		.500		**15 ± 0**			.500		**18 ± 0**
	4	.125		**15 ± 0**		4	.125		**18 ± 0**
		.250		**15 ± 0**			.250		**18 ± 0**
		.375		**15 ± 0**			.375		**18 ± 0**
		.500		**15 ± 0**			.500		**18 ± 0**
Peppers	1	.125	20 ± 0	**20 ± 0**	Parrots	1	.125	20 ± 0	**20 ± 0**
		.250		**20 ± 0**			.250		**20 ± 0**
		.375		**20 ± 0**			.375		**20 ± 0**
		.500		**20 ± 0**			.500		**20 ± 0**
	2	.125		**20 ± 0**		2	.125		**20 ± 0**
		.250		**20 ± 0**			.250		**20 ± 0**
		.375		**20 ± 0**			.375		**20 ± 0**
		.500		**20 ± 0**			.500		**20 ± 0**
	4	.125		21 ± 0		4	.125		**20 ± 0**
		.250		21 ± 0			.250		21 ± 0
		.375		**20 ± 0**			.375		**20 ± 0**
		.500		**20 ± 0**			.500		**20 ± 0**
Fish	1	.125	15 ± 0	**16 ± 0**	Pills	1	.125	19 ± 0	**19 ± 0**
		.250		**16 ± 0**			.250		**19 ± 0**
		.375		**16 ± 0**			.375		**19 ± 0**
		.500		**16 ± 0**			.500		**19 ± 0**
	2	.125		**16 ± 0**		2	.125		**19 ± 0**
		.250		**16 ± 0**			.250		**19 ± 0**
		.375		**16 ± 0**			.375		**19 ± 0**
		.500		**16 ± 0**			.500		**19 ± 0**
	4	.125		17 ± 0		4	.125		20 ± 0
		.250		**16 ± 0**			.250		20 ± 0
		.375		**16 ± 0**			.375		**19 ± 0**
		.500		**16 ± 0**			.500		**19 ± 0**

Table 6

MAE, 64 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	21 ± 0	**21 ± 0**	Goldhill	1	.125	12 ± 0	**12 ± 0**
		.250		**21 ± 0**			.250		**12 ± 0**
		.375		**21 ± 0**			.375		**12 ± 0**
		.500		**21 ± 0**			.500		**12 ± 0**
	2	.125		22 ± 0		2	.125		**12 ± 0**
		.250		**21 ± 0**			.250		**12 ± 0**
		.375		**21 ± 0**			.375		**12 ± 0**
		.500		**21 ± 0**			.500		**12 ± 0**
	4	.125		22 ± 0		4	.125		13 ± 0
		.250		22 ± 0			.250		**12 ± 0**
		.375		22 ± 0			.375		**12 ± 0**
		.500		22 ± 0			.500		**12 ± 0**
Lenna	1	.125	12 ± 0	**12 ± 0**	Motocross	1	.125	13 ± 0	**13 ± 0**
		.250		**12 ± 0**			.250		**13 ± 0**
		.375		**12 ± 0**			.375		**13 ± 0**
		.500		**12 ± 0**			.500		**13 ± 0**
	2	.125		**12 ± 0**		2	.125		**13 ± 0**
		.250		**12 ± 0**			.250		**13 ± 0**
		.375		**12 ± 0**			.375		**13 ± 0**
		.500		**12 ± 0**			.500		**13 ± 0**
	4	.125		**12 ± 0**		4	.125		14 ± 0
		.250		**12 ± 0**			.250		14 ± 0
		.375		**12 ± 0**			.375		**13 ± 0**
		.500		**12 ± 0**			.500		14 ± 0
Peppers	1	.125	16 ± 0	**16 ± 0**	Parrots	1	.125	15 ± 0	**15 ± 0**
		.250		**16 ± 0**			.250		**15 ± 0**
		.375		**16 ± 0**			.375		**15 ± 0**
		.500		**16 ± 0**			.500		**15 ± 0**
	2	.125		**16 ± 0**		2	.125		**15 ± 0**
		.250		**16 ± 0**			.250		**15 ± 0**
		.375		**16 ± 0**			.375		**15 ± 0**
		.500		**16 ± 0**			.500		**15 ± 0**
	4	.125		**16 ± 0**		4	.125		**15 ± 0**
		.250		**16 ± 0**			.250		**15 ± 0**
		.375		**16 ± 0**			.375		**15 ± 0**
		.500		**16 ± 0**			.500		**15 ± 0**
Fish	1	.125	12 ± 0	**12 ± 0**	Pills	1	.125	14 ± 0	**14 ± 0**
		.250		**12 ± 0**			.250		**14 ± 0**
		.375		**12 ± 0**			.375		**14 ± 0**
		.500		**12 ± 0**			.500		**14 ± 0**
	2	.125		**12 ± 0**		2	.125		**14 ± 0**
		.250		**12 ± 0**			.250		**14 ± 0**
		.375		**12 ± 0**			.375		**14 ± 0**
		.500		**12 ± 0**			.500		**14 ± 0**
	4	.125		13 ± 0		4	.125		15 ± 0
		.250		13 ± 0			.250		15 ± 0
		.375		13 ± 0			.375		**14 ± 0**
		.500		13 ± 0			.500		**14 ± 0**

Table 7

MAE, 128 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	17 ± 0	**17 ± 0**	Goldhill	1	.125	10 ± 0	**10 ± 0**
		.250		**17 ± 0**			.250		**10 ± 0**
		.375		**17 ± 0**			.375		**10 ± 0**
		.500		**17 ± 0**			.500		**10 ± 0**
	2	.125		**17 ± 0**		2	.125		**10 ± 0**
		.250		**17 ± 0**			.250		**10 ± 0**
		.375		**17 ± 0**			.375		**10 ± 0**
		.500		**17 ± 0**			.500		**10 ± 0**
	4	.125		18 ± 0		4	.125		**10 ± 0**
		.250		18 ± 0			.250		**10 ± 0**
		.375		18 ± 0			.375		**10 ± 0**
		.500		18 ± 0			.500		**10 ± 0**
Lenna	1	.125	9 ± 0	**9 ± 0**	Motocross	1	.125	10 ± 0	**10 ± 0**
		.250		**9 ± 0**			.250		**10 ± 0**
		.375		**9 ± 0**			.375		**10 ± 0**
		.500		**9 ± 0**			.500		**10 ± 0**
	2	.125		**9 ± 0**		2	.125		**10 ± 0**
		.250		**9 ± 0**			.250		**10 ± 0**
		.375		**9 ± 0**			.375		**10 ± 0**
		.500		**9 ± 0**			.500		**10 ± 0**
	4	.125		10 ± 0		4	.125		11 ± 0
		.250		10 ± 0			.250		**10 ± 0**
		.375		10 ± 0			.375		**10 ± 0**
		.500		**9 ± 0**			.500		**10 ± 0**
Peppers	1	.125	12 ± 0	**12 ± 0**	Parrots	1	.125	11 ± 0	**11 ± 0**
		.250		**12 ± 0**			.250		**11 ± 0**
		.375		**12 ± 0**			.375		**11 ± 0**
		.500		**12 ± 0**			.500		**11 ± 0**
	2	.125		**13 ± 0**		2	.125		12 ± 0
		.250		**12 ± 0**			.250		12 ± 0
		.375		**12 ± 0**			.375		**11 ± 0**
		.500		**12 ± 0**			.500		**11 ± 0**
	4	.125		13 ± 0		4	.125		12 ± 0
		.250		13 ± 0			.250		12 ± 0
		.375		13 ± 0			.375		12 ± 0
		.500		13 ± 0			.500		12 ± 0
Fish	1	.125	9 ± 0	10 ± 0	Pills	1	.125	11 ± 0	**11 ± 0**
		.250		**9 ± 0**			.250		**11 ± 0**
		.375		**9 ± 0**			.375		**11 ± 0**
		.500		**9 ± 0**			.500		**11 ± 0**
	2	.125		10 ± 0		2	.125		**11 ± 0**
		.250		10 ± 0			.250		**11 ± 0**
		.375		10 ± 0			.375		**11 ± 0**
		.500		10 ± 0			.500		**11 ± 0**
	4	.125		11 ± 0		4	.125		**11 ± 0**
		.250		10 ± 0			.250		**11 ± 0**
		.375		10 ± 0			.375		**11 ± 0**
		.500		10 ± 0			.500		**11 ± 0**

Table 8

MAE, 256 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	14 ± 0	**14 ± 0**	Goldhill	1	.125	8 ± 0	**8 ± 0**
		.250		**14 ± 0**			.250		**8 ± 0**
		.375		**14 ± 0**			.375		**8 ± 0**
		.500		**14 ± 0**			.500		**8 ± 0**
	2	.125		**14 ± 0**		2	.125		**8 ± 0**
		.250		**14 ± 0**			.250		**8 ± 0**
		.375		**14 ± 0**			.375		**8 ± 0**
		.500		**14 ± 0**			.500		**8 ± 0**
	4	.125		15 ± 0		4	.125		**8 ± 0**
		.250		**14 ± 0**			.250		**8 ± 0**
		.375		**14 ± 0**			.375		**8 ± 0**
		.500		**14 ± 0**			.500		**8 ± 0**
Lenna	1	.125	7 ± 0	8 ± 0	Motocross	1	.125	8 ± 0	**8 ± 0**
		.250		**7 ± 0**			.250		**8 ± 0**
		.375		**7 ± 0**			.375		**8 ± 0**
		.500		**7 ± 0**			.500		**8 ± 0**
	2	.125		8 ± 0		2	.125		**8 ± 0**
		.250		8 ± 0			.250		**8 ± 0**
		.375		8 ± 0			.375		**8 ± 0**
		.500		8 ± 0			.500		**8 ± 0**
	4	.125		8 ± 0		4	.125		**8 ± 0**
		.250		8 ± 0			.250		**8 ± 0**
		.375		8 ± 0			.375		**8 ± 0**
		.500		8 ± 0			.500		**8 ± 0**
Peppers	1	.125	10 ± 0	**10 ± 0**	Parrots	1	.125	8 ± 0	9 ± 0
		.250		**10 ± 0**			.250		**8 ± 0**
		.375		**10 ± 0**			.375		**8 ± 0**
		.500		**10 ± 0**			.500		**8 ± 0**
	2	.125		**10 ± 0**		2	.125		9 ± 0
		.250		**10 ± 0**			.250		9 ± 0
		.375		**10 ± 0**			.375		9 ± 0
		.500		**10 ± 0**			.500		9 ± 0
	4	.125		11 ± 0		4	.125		9 ± 0
		.250		**10 ± 0**			.250		9 ± 0
		.375		**10 ± 0**			.375		9 ± 0
		.500		**10 ± 0**			.500		9 ± 0
Fish	1	.125	7 ± 0	8 ± 0	Pills	1	.125	8 ± 0	**9 ± 0**
		.250		**7 ± 0**			.250		**9 ± 0**
		.375		**7 ± 0**			.375		**9 ± 0**
		.500		**7 ± 0**			.500		**9 ± 0**
	2	.125		8 ± 0		2	.125		**9 ± 0**
		.250		8 ± 0			.250		**9 ± 0**
		.375		8 ± 0			.375		**9 ± 0**
		.500		8 ± 0			.500		**9 ± 0**
	4	.125		9 ± 0		4	.125		**9 ± 0**
		.250		8 ± 0			.250		**9 ± 0**
		.375		8 ± 0			.375		**9 ± 0**
		.500		8 ± 0			.500		**9 ± 0**

Table 9

MSE, 32 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	378 ± 2	**379 ± 3**	Goldhill	1	.125	145 ± 2	146 ± 2
		.250		382 ± 5			.250		146 ± 3
		.375		**379 ± 3**			.375		**145 ± 1**
		.500		381 ± 4			.500		146 ± 2
	2	.125		385 ± 3		2	.125		147 ± 2
		.250		382 ± 5			.250		146 ± 2
		.375		382 ± 3			.375		146 ± 2
		.500		383 ± 3			.500		146 ± 1
	4	.125		398 ± 5		4	.125		151 ± 3
		.250		395 ± 5			.250		149 ± 3
		.375		390 ± 6			.375		148 ± 1
		.500		385 ± 3			.500		149 ± 3
Lenna	1	.125	120 ± 1	**120 ± 1**	Motocross	1	.125	194 ± 5	195 ± 6
		.250		121 ± 2			.250		194 ± 4
		.375		**120 ± 1**			.375		**192 ± 3**
		.500		**120 ± 1**			.500		195 ± 4
	2	.125		122 ± 1		2	.125		196 ± 2
		.250		122 ± 1			.250		198 ± 5
		.375		121 ± 2			.375		197 ± 5
		.500		121 ± 1			.500		194 ± 3
	4	.125		125 ± 1		4	.125		203 ± 6
		.250		124 ± 3			.250		199 ± 5
		.375		123 ± 2			.375		197 ± 6
		.500		123 ± 2			.500		199 ± 5
Peppers	1	.125	233 ± 4	234 ± 4	Parrots	1	.125	238 ± 4	239 ± 4
		.250		233 ± 3			.250		239 ± 5
		.375		234 ± 3			.375		239 ± 5
		.500		234 ± 2			.500		**238 ± 6**
	2	.125		235 ± 4		2	.125		242 ± 5
		.250		234 ± 3			.250		242 ± 5
		.375		233 ± 3			.375		241 ± 3
		.500		**232 ± 3**			.500		244 ± 6
	4	.125		244 ± 6		4	.125		247 ± 5
		.250		240 ± 4			.250		246 ± 6
		.375		235 ± 4			.375		246 ± 6
		.500		236 ± 2			.500		244 ± 7
Fish	1	.125	141 ± 2	143 ± 2	Pills	1	.125	203 ± 4	203 ± 3
		.250		**142 ± 2**			.250		204 ± 4
		.375		**142 ± 4**			.375		**202 ± 2**
		.500		**142 ± 3**			.500		206 ± 4
	2	.125		151 ± 3		2	.125		203 ± 2
		.250		147 ± 3			.250		205 ± 3
		.375		146 ± 3			.375		205 ± 6
		.500		144 ± 3			.500		205 ± 3
	4	.125		161 ± 5		4	.125		208 ± 4
		.250		153 ± 3			.250		208 ± 6
		.375		152 ± 5			.375		206 ± 3
		.500		148 ± 2			.500		206 ± 3

Table 10

MSE, 64 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	238 ± 1	240 ± 1	Goldhill	1	.125	85 ± 1	86 ± 1
		.250		**239 ± 2**			.250		**85 ± 1**
		.375		**239 ± 1**			.375		**85 ± 1**
		.500		**239 ± 1**			.500		**85 ± 0**
	2	.125		245 ± 1		2	.125		87 ± 1
		.250		243 ± 1			.250		86 ± 1
		.375		242 ± 1			.375		86 ± 1
		.500		242 ± 2			.500		**85 ± 0**
	4	.125		258 ± 3		4	.125		91 ± 1
		.250		251 ± 1			.250		88 ± 1
		.375		248 ± 3			.375		89 ± 1
		.500		248 ± 1			.500		89 ± 1
Lenna	1	.125	73 ± 1	**73 ± 1**	Motocross	1	.125	109 ± 1	**109 ± 1**
		.250		**73 ± 0**			.250		110 ± 2
		.375		**73 ± 1**			.375		**109 ± 1**
		.500		**73 ± 1**			.500		**109 ± 1**
	2	.125		75 ± 1		2	.125		112 ± 2
		.250		74 ± 1			.250		112 ± 1
		.375		74 ± 1			.375		111 ± 2
		.500		74 ± 1			.500		111 ± 2
	4	.125		79 ± 1		4	.125		117 ± 3
		.250		77 ± 1			.250		115 ± 2
		.375		76 ± 1			.375		112 ± 1
		.500		76 ± 1			.500		113 ± 1
Peppers	1	.125	137 ± 3	138 ± 3	Parrots	1	.125	128 ± 1	129 ± 1
		.250		138 ± 4			.250		**128 ± 1**
		.375		**136 ± 1**			.375		129 ± 2
		.500		**136 ± 3**			.500		**128 ± 1**
	2	.125		139 ± 3		2	.125		136 ± 3
		.250		138 ± 2			.250		136 ± 2
		.375		137 ± 1			.375		134 ± 1
		.500		137 ± 1			.500		136 ± 2
	4	.125		146 ± 2		4	.125		140 ± 3
		.250		144 ± 2			.250		139 ± 3
		.375		143 ± 1			.375		138 ± 2
		.500		141 ± 2			.500		137 ± 2
Fish	1	.125	85 ± 1	87 ± 1	Pills	1	.125	114 ± 1	114 ± 1
		.250		86 ± 1			.250		114 ± 1
		.375		86 ± 1			.375		115 ± 2
		.500		**85 ± 1**			.500		**113 ± 1**
	2	.125		92 ± 1		2	.125		115 ± 1
		.250		90 ± 2			.250		115 ± 1
		.375		88 ± 1			.375		115 ± 1
		.500		87 ± 1			.500		114 ± 1
	4	.125		102 ± 2		4	.125		121 ± 2
		.250		96 ± 1			.250		117 ± 1
		.375		96 ± 2			.375		116 ± 1
		.500		94 ± 1			.500		117 ± 2

Table 11

MSE, 128 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	152 ± 1	155 ± 1	Goldhill	1	.125	53 ± 0	**53 ± 0**
		.250		154 ± 0			.250		**53 ± 0**
		.375		**153 ± 1**			.375		**53 ± 0**
		.500		**153 ± 1**			.500		**53 ± 0**
	2	.125		160 ± 1		2	.125		55 ± 0
		.250		158 ± 1			.250		54 ± 1
		.375		157 ± 1			.375		54 ± 0
		.500		157 ± 1			.500		54 ± 0
	4	.125		172 ± 3		4	.125		58 ± 1
		.250		167 ± 1			.250		56 ± 0
		.375		164 ± 1			.375		56 ± 1
		.500		162 ± 1			.500		55 ± 1
Lenna	1	.125	47 ± 0	**47 ± 0**	Motocross	1	.125	63 ± 1	**63 ± 1**
		.250		**47 ± 0**			.250		**63 ± 1**
		.375		**47 ± 0**			.375		**63 ± 0**
		.500		**47 ± 0**			.500		**63 ± 0**
	2	.125		49 ± 0		2	.125		66 ± 1
		.250		48 ± 0			.250		65 ± 0
		.375		48 ± 0			.375		65 ± 0
		.500		**47 ± 0**			.500		65 ± 1
	4	.125		52 ± 0		4	.125		70 ± 1
		.250		51 ± 0			.250		68 ± 1
		.375		50 ± 0			.375		67 ± 1
		.500		49 ± 0			.500		67 ± 1
Peppers	1	.125	84 ± 1	85 ± 1	Parrots	1	.125	73 ± 1	74 ± 1
		.250		85 ± 0			.250		**73 ± 1**
		.375		**84 ± 0**			.375		**73 ± 1**
		.500		**84 ± 1**			.500		**73 ± 1**
	2	.125		88 ± 1		2	.125		81 ± 1
		.250		87 ± 0			.250		81 ± 1
		.375		86 ± 1			.375		79 ± 1
		.500		86 ± 1			.500		79 ± 1
	4	.125		94 ± 1		4	.125		85 ± 2
		.250		92 ± 1			.250		82 ± 1
		.375		90 ± 1			.375		82 ± 2
		.500		89 ± 1			.500		81 ± 1
Fish	1	.125	52 ± 0	54 ± 0	Pills	1	.125	67 ± 0	**67 ± 0**
		.250		54 ± 1			.250		**67 ± 0**
		.375		**53 ± 0**			.375		**67 ± 1**
		.500		**53 ± 0**			.500		**67 ± 1**
	2	.125		59 ± 1		2	.125		69 ± 1
		.250		56 ± 1			.250		68 ± 1
		.375		56 ± 1			.375		68 ± 1
		.500		55 ± 1			.500		68 ± 1
	4	.125		70 ± 2		4	.125		73 ± 1
		.250		63 ± 1			.250		71 ± 1
		.375		62 ± 1			.375		70 ± 1
		.500		60 ± 1			.500		70 ± 0

Table 12

MSE, 256 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	97 ± 0	100 ± 0	Goldhill	1	.125	34 ± 0	35 ± 0
		.250		99 ± 0			.250		**34 ± 0**
		.375		**98 ± 0**			.375		**34 ± 0**
		.500		**98 ± 0**			.500		**34 ± 0**
	2	.125		105 ± 1		2	.125		36 ± 0
		.250		103 ± 0			.250		36 ± 0
		.375		102 ± 0			.375		35 ± 0
		.500		101 ± 0			.500		35 ± 0
	4	.125		114 ± 1		4	.125		39 ± 0
		.250		109 ± 1			.250		38 ± 0
		.375		107 ± 0			.375		37 ± 0
		.500		107 ± 1			.500		37 ± 0
Lenna	1	.125	30 ± 0	**31 ± 0**	Motocross	1	.125	37 ± 0	38 ± 0
		.250		**31 ± 0**			.250		38 ± 0
		.375		**31 ± 0**			.375		38 ± 0
		.500		**31 ± 0**			.500		38 ± 0
	2	.125		33 ± 0		2	.125		40 ± 0
		.250		32 ± 0			.250		40 ± 0
		.375		**31 ± 0**			.375		39 ± 0
		.500		**31 ± 0**			.500		39 ± 0
	4	.125		35 ± 0		4	.125		44 ± 0
		.250		34 ± 0			.250		42 ± 0
		.375		33 ± 0			.375		42 ± 0
		.500		33 ± 0			.500		41 ± 0
Peppers	1	.125	54 ± 0	55 ± 0	Parrots	1	.125	42 ± 0	**43 ± 0**
		.250		55 ± 0			.250		**43 ± 0**
		.375		**54 ± 0**			.375		**43 ± 0**
		.500		**54 ± 0**			.500		**43 ± 0**
	2	.125		58 ± 0		2	.125		49 ± 0
		.250		57 ± 0			.250		49 ± 0
		.375		56 ± 0			.375		48 ± 1
		.500		56 ± 0			.500		48 ± 0
	4	.125		63 ± 1		4	.125		53 ± 1
		.250		61 ± 1			.250		51 ± 1
		.375		60 ± 0			.375		50 ± 0
		.500		59 ± 1			.500		49 ± 1
Fish	1	.125	32 ± 0	34 ± 0	Pills	1	.125	41 ± 0	42 ± 0
		.250		**33 ± 0**			.250		42 ± 0
		.375		**33 ± 0**			.375		42 ± 0
		.500		**33 ± 0**			.500		**41 ± 0**
	2	.125		38 ± 0		2	.125		44 ± 0
		.250		36 ± 0			.250		43 ± 0
		.375		35 ± 0			.375		42 ± 0
		.500		35 ± 0			.500		42 ± 0
	4	.125		52 ± 2		4	.125		47 ± 0
		.250		42 ± 1			.250		45 ± 0
		.375		40 ± 1			.375		45 ± 0
		.500		40 ± 0			.500		44 ± 0

Table 13

Time, 32 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	443 ± 70	5538 ± 272	Goldhill	1	.125	738 ± 231	12125 ± 620
		.250		11106 ± 572			.250		25034 ± 998
		.375		15828 ± 364			.375		35611 ± 1588
		.500		21066 ± 1658			.500		47207 ± 3637
	2	.125		368 ± 19		2	.125		838 ± 46
		.250		696 ± 18			.250		1680 ± 254
		.375		1006 ± 14			.375		2325 ± 76
		.500		1370 ± 29			.500		3177 ± 143
	4	.125		**27 ± 5**		4	.125		**63 ± 4**
		.250		55 ± 4			.250		117 ± 9
		.375		87 ± 8			.375		192 ± 37
		.500		116 ± 8			.500		230 ± 30
Lenna	1	.125	477 ± 62	5640 ± 143	Motocross	1	.125	604 ± 106	9738 ± 206
		.250		11411 ± 253			.250		19341 ± 263
		.375		16548 ± 216			.375		28962 ± 405
		.500		22776 ± 692			.500		38490 ± 487
	2	.125		400 ± 53		2	.125		648 ± 13
		.250		841 ± 152			.250		1283 ± 28
		.375		1114 ± 55			.375		1894 ± 34
		.500		1499 ± 104			.500		2533 ± 65
	4	.125		**31 ± 5**		4	.125		**53 ± 3**
		.250		64 ± 18			.250		99 ± 5
		.375		90 ± 10			.375		146 ± 9
		.500		121 ± 25			.500		185 ± 6
Peppers	1	.125	474 ± 122	5782 ± 83	Parrots	1	.125	559 ± 124	11221 ± 205
		.250		10630 ± 666			.250		22553 ± 407
		.375		15028 ± 102			.375		33761 ± 643
		.500		20661 ± 968			.500		45225 ± 900
	2	.125		356 ± 12		2	.125		759 ± 18
		.250		684 ± 10			.250		1476 ± 24
		.375		1026 ± 60			.375		2215 ± 40
		.500		1327 ± 11			.500		2912 ± 53
	4	.125		**28 ± 3**		4	.125		**61 ± 5**
		.250		52 ± 6			.250		114 ± 3
		.375		88 ± 9			.375		163 ± 6
		.500		102 ± 10			.500		214 ± 12
Fish	1	.125	100 ± 18	311 ± 41	Pills	1	.125	647 ± 117	12416 ± 121
		.250		581 ± 55			.250		24800 ± 142
		.375		813 ± 23			.375		36970 ± 332
		.500		1064 ± 22			.500		49485 ± 232
	2	.125		26 ± 4		2	.125		825 ± 18
		.250		44 ± 3			.250		1634 ± 26
		.375		66 ± 4			.375		2413 ± 21
		.500		83 ± 5			.500		3201 ± 35
	4	.125		**3 ± 3**		4	.125		**68 ± 4**
		.250		4 ± 1			.250		122 ± 6
		.375		6 ± 2			.375		182 ± 11
		.500		6 ± 2			.500		233 ± 14

Table 14

Time, 64 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	849 ± 100	5486 ± 376	Goldhill	1	.125	1454 ± 276	13048 ± 1162
		.250		10650 ± 342			.250		24050 ± 519
		.375		15764 ± 464			.375		36495 ± 1075
		.500		21522 ± 564			.500		49039 ± 2453
	2	.125		382 ± 16		2	.125		850 ± 67
		.250		754 ± 50			.250		1688 ± 86
		.375		1070 ± 22			.375		2432 ± 107
		.500		1436 ± 42			.500		3170 ± 192
	4	.125		**34 ± 3**		4	.125		**74 ± 4**
		.250		64 ± 4			.250		134 ± 8
		.375		95 ± 5			.375		190 ± 8
		.500		123 ± 9			.500		261 ± 21
Lenna	1	.125	884 ± 159	5720 ± 189	Motocross	1	.125	1094 ± 173	9710 ± 59
		.250		11795 ± 445			.250		19274 ± 145
		.375		17390 ± 535			.375		28926 ± 385
		.500		22569 ± 640			.500		38677 ± 397
	2	.125		427 ± 34		2	.125		676 ± 6
		.250		798 ± 27			.250		1314 ± 15
		.375		1148 ± 56			.375		1948 ± 22
		.500		1513 ± 28			.500		2589 ± 40
	4	.125		**34 ± 5**		4	.125		**61 ± 3**
		.250		71 ± 9			.250		111 ± 4
		.375		94 ± 7			.375		155 ± 5
		.500		120 ± 8			.500		207 ± 8
Peppers	1	.125	842 ± 139	5164 ± 45	Parrots	1	.125	1190 ± 316	11643 ± 83
		.250		10376 ± 358			.250		23062 ± 183
		.375		16934 ± 684			.375		34618 ± 529
		.500		21577 ± 573			.500		45950 ± 557
	2	.125		509 ± 107		2	.125		798 ± 12
		.250		759 ± 79			.250		1561 ± 26
		.375		1076 ± 49			.375		2294 ± 43
		.500		1418 ± 48			.500		3050 ± 63
	4	.125		**36 ± 6**		4	.125		**68 ± 4**
		.250		67 ± 5			.250		124 ± 6
		.375		93 ± 4			.375		181 ± 7
		.500		129 ± 10			.500		236 ± 9
Fish	1	.125	191 ± 29	317 ± 25	Pills	1	.125	1334 ± 169	12666 ± 81
		.250		582 ± 21			.250		25170 ± 112
		.375		862 ± 23			.375		37662 ± 103
		.500		1134 ± 28			.500		50288 ± 285
	2	.125		26 ± 3		2	.125		871 ± 12
		.250		51 ± 3			.250		1697 ± 20
		.375		75 ± 3			.375		2504 ± 35
		.500		98 ± 7			.500		3313 ± 32
	4	.125		**3 ± 3**		4	.125		**74 ± 4**
		.250		6 ± 2			.250		138 ± 6
		.375		8 ± 3			.375		196 ± 4
		.500		10 ± 3			.500		252 ± 8

Table 15

Time, 128 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	1706 ± 252	5666 ± 158	Goldhill	1	.125	2285 ± 365	12735 ± 643
		.250		10961 ± 322			.250		25234 ± 1234
		.375		15998 ± 422			.375		34962 ± 1252
		.500		21217 ± 537			.500		47425 ± 3080
	2	.125		434 ± 11		2	.125		847 ± 28
		.250		814 ± 30			.250		1597 ± 28
		.375		1205 ± 50			.375		2326 ± 47
		.500		1584 ± 103			.500		3098 ± 51
	4	.125		**47 ± 6**		4	.125		**83 ± 5**
		.250		79 ± 8			.250		145 ± 7
		.375		116 ± 8			.375		201 ± 6
		.500		156 ± 19			.500		263 ± 9
Lenna	1	.125	1657 ± 206	5969 ± 186	Motocross	1	.125	2217 ± 408	9903 ± 69
		.250		11428 ± 344			.250		19446 ± 116
		.375		17533 ± 605			.375		29122 ± 311
		.500		23258 ± 581			.500		38910 ± 204
	2	.125		488 ± 74		2	.125		742 ± 28
		.250		836 ± 22			.250		1399 ± 27
		.375		1272 ± 112			.375		2053 ± 27
		.500		1626 ± 67			.500		2740 ± 53
	4	.125		**45 ± 5**		4	.125		**72 ± 5**
		.250		89 ± 18			.250		132 ± 9
		.375		126 ± 17			.375		179 ± 11
		.500		150 ± 10			.500		238 ± 14
Peppers	1	.125	1598 ± 169	5722 ± 410	Parrots	1	.125	2270 ± 279	11954 ± 163
		.250		10489 ± 272			.250		23661 ± 284
		.375		16254 ± 1060			.375		35184 ± 312
		.500		20026 ± 154			.500		46758 ± 468
	2	.125		411 ± 17		2	.125		872 ± 29
		.250		881 ± 164			.250		1642 ± 32
		.375		1109 ± 21			.375		2431 ± 47
		.500		1461 ± 21			.500		3202 ± 49
	4	.125		**46 ± 3**		4	.125		**79 ± 3**
		.250		76 ± 6			.250		150 ± 7
		.375		118 ± 11			.375		207 ± 7
		.500		146 ± 9			.500		272 ± 12
Fish	1	.125	396 ± 66	381 ± 78	Pills	1	.125	2492 ± 267	13048 ± 115
		.250		628 ± 24			.250		25680 ± 71
		.375		971 ± 67			.375		38374 ± 190
		.500		1189 ± 36			.500		51088 ± 259
	2	.125		31 ± 5		2	.125		950 ± 23
		.250		63 ± 11			.250		1805 ± 21
		.375		92 ± 7			.375		2649 ± 31
		.500		127 ± 14			.500		3484 ± 45
	4	.125		**4 ± 3**		4	.125		**88 ± 4**
		.250		6 ± 2			.250		157 ± 9
		.375		8 ± 4			.375		225 ± 8
		.500		13 ± 4			.500		286 ± 8

Table 16

Time, 256 Colors

IMG	DF	CF	KM	CM	IMG	DF	CF	KM	CM
Baboon	1	.125	3152 ± 184	6252 ± 326	Goldhill	1	.125	4240 ± 315	11930 ± 129
		.250		11285 ± 223			.250		23373 ± 301
		.375		15921 ± 322			.375		34662 ± 411
		.500		22468 ± 433			.500		45838 ± 343
	2	.125		522 ± 42		2	.125		942 ± 15
		.250		898 ± 19			.250		1766 ± 31
		.375		1334 ± 46			.375		2599 ± 64
		.500		1796 ± 115			.500		3374 ± 37
	4	.125		**58 ± 7**		4	.125		**104 ± 5**
		.250		105 ± 8			.250		175 ± 5
		.375		152 ± 9			.375		248 ± 9
		.500		190 ± 14			.500		326 ± 14
Lenna	1	.125	3026 ± 208	6118 ± 49	Motocross	1	.125	4236 ± 505	10146 ± 85
		.250		11749 ± 76			.250		19765 ± 159
		.375		19255 ± 1268			.375		29367 ± 217
		.500		26009 ± 404			.500		39146 ± 497
	2	.125		575 ± 27		2	.125		830 ± 15
		.250		1047 ± 22			.250		1553 ± 43
		.375		1511 ± 48			.375		2207 ± 33
		.500		2137 ± 207			.500		2909 ± 46
	4	.125		**70 ± 16**		4	.125		**92 ± 4**
		.250		111 ± 8			.250		162 ± 6
		.375		158 ± 15			.375		219 ± 9
		.500		205 ± 8			.500		294 ± 17
Peppers	1	.125	2929 ± 227	5570 ± 58	Parrots	1	.125	4026 ± 379	12389 ± 115
		.250		10691 ± 127			.250		24471 ± 261
		.375		15789 ± 212			.375		36392 ± 364
		.500		20985 ± 284			.500		48088 ± 329
	2	.125		490 ± 9		2	.125		975 ± 13
		.250		890 ± 18			.250		1809 ± 28
		.375		1277 ± 32			.375		2645 ± 78
		.500		1673 ± 35			.500		3484 ± 75
	4	.125		**58 ± 5**		4	.125		**105 ± 5**
		.250		98 ± 6			.250		174 ± 7
		.375		146 ± 13			.375		253 ± 14
		.500		186 ± 5			.500		329 ± 10
Fish	1	.125	681 ± 64	406 ± 11	Pills	1	.125	4606 ± 348	13482 ± 113
		.250		755 ± 25			.250		26472 ± 141
		.375		1049 ± 33			.375		39570 ± 207
		.500		1554 ± 261			.500		52404 ± 347
	2	.125		52 ± 13		2	.125		1046 ± 16
		.250		84 ± 7			.250		1989 ± 29
		.375		118 ± 5			.375		2870 ± 36
		.500		158 ± 17			.500		3830 ± 55
	4	.125		**6 ± 3**		4	.125		**108 ± 5**
		.250		10 ± 4			.250		199 ± 10
		.375		15 ± 3			.375		276 ± 11
		.500		18 ± 5			.500		364 ± 15

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, coremeans was introduced as a fast and effective quantization method. First, a data subsampling procedure was introduced as means of data reduction. Then, an effective initialization method, k-means++ was presented with the objective of selecting better initial centers. Finally, the coreset construction algorithm was outlined as the main performance enhancing component of coremeans. The proposed method was implemented and a comprehensive experiment conducted on a diverse set of test images commonly used in the color quantization literature. The experimental results demonstrated that the proposed method outperforms k-means in terms of speed, achieving speedups of up to 100 times when run with suitable parameter values. The proposed method gives comparable-quality results to k-means.

The proposed method was tested against k-means only as the former is a modified version of the latter. Additionally, Celebi *et al.* (2015) conducted extensive experiments where they compared the quality and speed of a large number of popular color quantization methods including popularity, median-cut, modified popularity, octree, variance-based method, greedy orthogonal bipartitioning method, center-cut, self-organizing map, radius-weighted mean-cut, modified maximin, pairwise clustering, split and merge, Cheng & Yang, adaptive distributing units, and weighted sort-means. They found that k-means outperformed the rival methods by a large margin in nearly every case. The proposed coremeans method is significantly faster than k-means and the two methods give very similar results. This means that coremeans will also outperform the large number of color quantization methods tested by Celebi *et al.* (2015).

In the experiment conducted, only k-means++ was implemented as the initialization method. Numerous other initialization methods exist (Celebi *et al.*, 2013), and using them in combination with coresets could provide some interesting results. Furthermore, a comparison between the results obtained by other quantization methods that use a coreset as a source could provide valuable insight into practical coreset use. Finally, acceleration of the proposed coremeans method could also be investigated, for example, using the recent approach proposed by Bachem *et al.* (2016).

REFERENCES

Arthur, D. & Vassilvitskii, S. (2007). K-means++: The Advantages of Careful Seeding. *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035. doi: 10.1145/1283383.1283494

Bachem, O., Lucic, M., & Krause, A. (2017). *Practical Coreset Construction for Machine Learning.* Retrieved October 27, 2017 from the arXiv website: https://arxiv.org/abs/1703.06476

Bachem, O., Lucic, M., Hassan, S. H., & Krause, A. (2016). Fast and Provably Good Seedings for k-Means. *Proceedings of the 2016 Neural Information Processing Systems Conference*, 55–63. Retrieved October 27, 2017 from https://papers.nips.cc/paper/6478-fast-and-provably-good-seedings-for-k-means

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York, NY: Springer.

Blum, C., & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computer Surveys*, *35*(3), 268–308. doi: 10.1145/937503.937505

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.

Burge, M. J., & Burger, W. (2009). *Principles of Digital Image Processing: Core*

Algorithms. London, UK: Springer.

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., Wen, Q., & Hwang, S. (2015). An Effective Real-Time Color Quantization Methods Based on Divisive Hierarchical Clustering. *Journal of Real-Time Image Processing, 10*(2), 329–344. doi: 10.1007/s11554-012-0291-4

Cheng, S., & Yang, C. (2001). Fast and Novel Technique for Color Quantization Using Reduction of Color Space Dimensionality. *Pattern Recognition Letters, 22*(8), 845–856. doi: 10.1016/S0167-8655(01)00025-3

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

Feldman, D., Ozer, S., & Rus, D. (2017). Coresets for Vector Summarization with Applications to Network Graphs. *Proceedings of the 34th International Conference on Machine Learning,* 1117–1125.

Feldman, D., Schmidt, M., & Sohler, C. (2013). Turning Big Data into Tiny Data: Constant-Size Coresets for K-Means, PCA and Projective Clustering. *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1434–1453. doi: 10.1137/1.9781611973105.103

Gentile, R. S., Allebach, J. P., & Walowit, E. (1990). Quantization of Color Images Based on Uniform Color Spaces. *Journal of Imaging Technology*, 16(1), 11–21. doi: 10.1145/965145.801294

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

Har-Peled, S. (2011). *Geometric Approximation Algorithms*. Providence, RI: American Mathematical Society.

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

Hu, Z., Su, Q., & Xia, X. (2016). Multiobjective Image Color Quantization Algorithm Based on Self-Adaptive Hybrid Differential Evolution. *Computational Intelligence and Neuroscience, 2016*. doi: 10.1155/2016/2450431

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

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

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

Schaefer, G., & Nolle, L. (2015). A Hybrid Color Quantization Algorithm Incorporating a Human Visual Perception Model. *Computational Intelligence, 31*(4), 684-698. doi: 10.1111/coin.12043

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

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

Velho, L., Gomez, J., & Sobreiro, M. V. R. (1997). Color Image Quantization by Pairwise Clustering. *Proceedings of the 10th Brazilian Symposium on Computer*

Graphics and Image Processing, 203–210. doi: 10.1109/SIGRA.1997.625178

Volkov, M. (2016). *Machine Learning and Coresets for Automated Real-Time Data Segmentation and Summarization* (Doctoral dissertation). Retrieved from DSpace@MIT. (http://hdl.handle.net/1721.1/107865)

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

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

Ward, J. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association, 58*(301), 236–244. doi: 10.1080/01621459.1963.10500845

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, X. (1991). Efficient Statistical Computations for Optimal Color Quantization. In J. Arvo (Ed.), *Graphics Gems Volume II* (pp. 126–133). San Diego, CA: Academic Press.

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

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