# THE INCREMENTAL ONLINE K-MEANS CLUSTERING ALGORITHM AND ITS APPLICATION TO COLOR QUANTIZATION

by

Amber Abernathy

A thesis presented to the Department of Computer Science and Engineering
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 2022

## ACKNOWLEDGEMENT

First and foremost, I would like to express my sincerest thanks to my thesis and research advisor Dr. Emre Celebi, professor and chairperson of the Department of Computer Science and Engineering at the University of Central Arkansas. This research would not have been possible without his dedication, leadership, and support.

I would also like to thank the members of my thesis committee, Dr. Yu Sun and Dr. Sinan Kockara, for their support and positive experiences throughout my entire educational journey at the University of Central Arkansas.

Finally, I would like to thank the remaining faculty of the Department of Computer Science and Engineering for their time, influence, and support. Everyone has played a major role in both influencing my future and education, all while engraving positive memories along the way. Thank you all for all you do.

## VITA

Amber Abernathy was raised in Morley, Missouri before moving to Little Rock, Arkansas at the age of 5. In 2017, she graduated as class Valedictorian from Mayflower High School. After high school, she attended the University of Central Arkansas and graduated in May 2021 with a Bachelor of Science in Computer Science and minors in Mathematics and Business Administration.

## ABSTRACT

Color quantization (CQ) is a common image processing operation with various applications in computer graphics, image processing, and computer vision. CQ is essentially a large-scale combinatorial optimization problem in low dimensions. Many clustering algorithms, both of hierarchical and partitional types, have been applied to the CQ problem since the 1980s. In general, hierarchical CQ algorithms are faster, whereas partitional ones produce better results provided that they are initialized properly. In this thesis, we propose a novel partitional CQ algorithm based on a binary splitting formulation of MacQueen's online k-means algorithm. Experiments on a diverse set of public test images demonstrate that the proposed algorithm is significantly faster than two popular batch k-means algorithms while yielding nearly identical results. In addition, unlike MacQueen's original algorithm, the proposed algorithm is both deterministic and free of initialization. The presented algorithm may be of independent interest as a general-purpose clustering algorithm.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1 INTRODUCTION

True-color images have become truly ubiquitous over the past two decades. A typical true-color image may contain hundreds of thousands of colors, which makes it challenging to display, store, transmit, process, and analyze such an image. Color Quantization (CQ) is a common image processing operation, which aims to reduce the number of distinct colors in a true-color image with minimal distortion. Thus, CQ is fundamentally a large-scale combinatorial optimization problem in low dimensions. Recent applications of CQ include compression, segmentation, text localization/detection, color analysis, watermarking, non- photorealistic rendering, and content-based retrieval (Thompson, Celebi, & Buck 2020).

CQ consists of two main phases: palette design (the selection of a small set of colors that represents the colors in the input image) and pixel mapping (the assignment of each pixel in the input image to one of the palette colors). Since pixel mapping can be accomplished using a straightforward linear-time algorithm that maps each input pixel to the nearest palette color, most CQ studies deal with the computationally difficult palette design phase.

Many clustering algorithms, both of hierarchical and partitional types, have been applied to the palette design problem since the 1980s (Brun & Tremeau 2002). A hierarchical algorithm partitions the input data set into a set of nested clusters that are organized as a tree. These algorithms recursively find nested clusters in a top-down (divisive) or bottom-up (agglomerative) manner (Jain, Murty, & Flynn 1999). A partitional algorithm, on the other hand, partitions the input data set into a number of mutually exclusive subsets. These algorithms find all clusters simultaneously without

imposing a hierarchical structure (Jain, Murty, & Flynn 1999). In the context of CQ, hierarchical design algorithms are generally faster, whereas partitional ones often produce better results as long that they are initialized properly. Classic hierarchical CQ algorithms include median-cut (Heckbert 1982), octree (Gervautz & Purgathofer 1988), Wan *et al.*'s algorithm (Wan, Prusinkiewicz, & Wong 1990), Wu's algorithm (Wu 1991), Orchard & Bouman's algorithm (Orchard & Bouman 1991), center-cut (Joy & Xiang 1993), and rwm cut (Yang & Lin 1996). Partitional algorithms adapted to CQ include competitive learning (Celebi, Hwang, & Wen 2014), fuzzy c-means (Schaefer & Zhou 2009; Wen & Celebi 2011), k-means (Hu & Lee 2007; Hu & Su 2008) (Celebi 2009; Celebi 2011; Valenzuela, Celebi, & Schaefer 2018; Thompson, Celebi, & Buck 2020), k-harmonic means (Frackiewicz & Palus 2011), maximin (Xiang 1997), rough c-means (Schaefer, HU, Zhou, Peters, & Hassanien 2012), and self-organizing maps (Dekker 1994; Pei & Lo 1998; Chang, Xu, Xiao, & Srikanthan 2005; Wang, Lee, & Hsieh 2007; Rasti, Monadjemi, & Vafaei 2011).

In this thesis, we present a novel partitional CQ algorithm based on an online formulation of the celebrated k-means algorithm. The remainder of the thesis is organized as follows. Chapter 2 describes three known variants of the k-means clustering algorithm and the proposed fourth variant. Chapter 3 presents the experimental results and compares the proposed algorithm to conventional as well as state-of-the-art CQ algorithms. Finally, Chapter 4 gives the conclusions.

# CHAPTER 2 K-MEANS AND ITS VARIANTS

This chapter describes four variants of the k-means clustering algorithm: batch k-means, incremental batch k-means, online k-means, and incremental online k-means. The first three algorithms are known, and the last one is new.

## Batch K-Means

Batch k-means (BKM) (Forgy 1965), also known as Lloyd's algorithm (Lloyd 1982), is the most widely used partitional clustering algorithm (Wu, Kumar, Quinlan, et al. 2008). Given a data set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$ and a positive integer $K > 1$, BKM divides $X$ into $K$ mutually exclusive and exhaustive clusters $\{P_1, \ldots, P_K\}$, where each cluster $P_i$ is represented by a center $\boldsymbol{c}_i$. The algorithm starts with an arbitrary set of initial centers, customarily chosen uniformly at random from $X$. Each iteration is composed of two steps: assignment and update. In the assignment step, each data point is assigned to the nearest center. In the update step, each center is updated to be the centroid of the data points assigned to it. Each iteration can be shown to either decrease the Sum of Squared Error (SSE) defined as $\text{SSE} = \sum_{\mathbf{x} \in X} d_{\text{SE}}(\mathbf{x}, \{\mathbf{c}_1, \ldots, \mathbf{c}_K\})$ , where $d_{\text{SE}}(\mathbf{x}, C)$ denotes the squared Euclidean ($\ell_2^2$) dissimilarity between data point $\mathbf{x}$ and the nearest center in $C$, or leave it unchanged (at which point the algorithm is considered to have converged). Banerjee *et al.* (Banerjee, Merugu, Dhillon, Ghosh, & Lafferty 2005) proved that the optimal center of a cluster is given by the centroid of the cluster *only* for Bregman divergences (Bregman 1967), a family of nonmetric dissimilarity functions that includes the ($\ell_2^2$) dissimilarity, squared Mahalanobis dissimilarity, Kullback–Leibler divergence, and Itakura–Saito divergence.

Let $I$ be the input image in a CQ application. The data set $X$ then represents the pixels of $I$, $N$ is the number of pixels in $I$, $D$ is the number of color channels ($D = 3$ for the RGB color model), and $K$ is the number of desired colors in the output (quantized) image. The pseudocode for BKM is given in Table 1.

Table 1. *BKM Algorithm*

| Step | Description |
|---|---|
| 1 | Initialize the cluster centers $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$. |
| 2 | Assign each $\mathbf{x} \in X$ to the nearest center, *i.e.*, $\boldsymbol{c}_i$ with $i = \underset{k \in \{1,\ldots,K\}}{\arg min} \|\mathbf{x} - \mathbf{c}_k\|_2^2$, where $\|\cdot\|_2^2$ denotes the $\ell_2^2$ norm. |
| 3 | Update each cluster center $\mathbf{c}_i$ to be the centroid of the data points assigned to it, *i.e.*, $\mathbf{c}_i = (1/n_i) \sum_{\mathbf{x} \in P_i} \mathbf{x}$, where $n_i$ is the number of data points assigned to $\mathbf{c}_i$. |
| 4 | Repeat steps (2) and (3) until convergence. |

BKM converges to a local minimum of its objective when the cluster memberships of the data points stabilize (or, equivalently, when the cluster centers stabilize). Step (1), initialization, is the most important step as the algorithm is highly sensitive to the initialization of the centers. Adverse effects of poor initialization include empty clusters, slower convergence, and a higher chance of getting stuck in a bad local optimum (Celebi, Kingravi, & Vela 2013). Although it has a linear time complexity (in *N*, *D*, and *K*), BKM is computationally demanding to do its iterative nature (Celebi 2011). In general, the number of iterations cannot be predicted in advance and depends on the

number, dimensionality, and distribution of the data points as well as the number of clusters sought.

## Incremental Batch K-Means

Incremental batch k-means (IBKM) (Linde, Buzo, & Gray 1980) is a variant of BKM that features a built-in initialization scheme. The original BKM algorithm assumes that the algorithm is supplied with appropriate initial centers. IBKM, on the other hand, starts with a single center and incrementally adds new centers until the number of centers reaches K. The pseudocode for IBKM is given in Table 2.

Table 2. *IBKM Algorithm*

| Step | Description |
|---|---|
| 1 | Set $\mathbf{c}_0 = centroid(X)$ and iteration counter $t = 0$. |
| 2 | For each $k \in \{2^t - 1, \ldots, 2^{t+1} - 2\}$, split node $\mathbf{c}_k$ into nodes $\mathbf{c}_{2k+1}$ and $\mathbf{c}_{2k+2}$. |
| 3 | Refine centers $(2^{t+1} - 1), \ldots, (2^{t+2} - 2)$ using BKM (the entire data set $X$ is clustered.) |
| 4 | Set $t = t + 1$ and repeat steps (2) and (3) until $t = \log_2 K$. |

The algorithm begins by setting $\mathbf{c}_0$ to the centroid of $X$ (for $K = 1$, this choice of $\mathbf{c}_0$ is clearly optimal). It then adds $2^{t+1}$ new centers in iteration $t$ ($t \in \{0, \ldots, \log_2 K - 1\}$) splitting each of the existing centers into two. When a center $\mathbf{c}_k$ is split, the left child inherits its parent's attributes (*i.e.*, $\mathbf{c}_{2k+1} = \mathbf{c}_k$), whereas the right child becomes a slightly perturbed version of its parent (*i.e.*, $\mathbf{c}_{2k+1} = \mathbf{c}_k + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is an arbitrary

vector of small positive $\ell_2$ norm). Preserving the parent's attributes in the next iteration ensures that the SSE will not increase (Gray 1984). Note that in the IBKM pseudocode above, for simplicity, we assumed that $K$ is a power of two. If this is not the case, we perform $\lfloor \log_2 K \rfloor$ iterations as described above and then perform one last iteration in which we split only $K - 2^{\lfloor \log_2 K \rfloor}$ of the centers from the previous iteration.

While BKM is well known in virtually all scientific disciplines where clustering is employed (Celebi 2014), IBKM appears to be known primarily in the vector quantization literature from which it originated.

## Online K-Means

Online k-means (OKM) (MacQueen 1967), also known as sequential k-means or Mac-Queen's k-means, is an online variant of BKM. The two algorithms differ in how often and how they update the cluster centers. BKM updates all $K$ cluster centers at once after all data points are assigned to their respective nearest centers. OKM, on the other hand, updates a single cluster center immediately after a data point is assigned to it. OKM is considered a noisy version of BKM where the noise aids the algorithm in escaping bad local optima (Bottou 1995). The pseudocode for OKM is given in Table 3.

Table 3. *OKM Algorithm*

| Step | Description |
|---|---|
| 1 | Initialize the cluster centers $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$. |
| 2 | Set $n_1 = \cdots = n_K = 0$, where $n_i$ is the number of data points assigned to center $\mathbf{c}_i$. |
| 3 | Select a data point $\mathbf{x}$ from $X$ uniformly at random. |
| 4 | Assign $\mathbf{x}$ to the nearest center, say $\mathbf{c}_i$ (refer to step (2) in the BKM pseudocode). |
| 5 | Increment $n_i$ $(i.e., n_i = n_i + 1)$ and then update $\mathbf{c}_i$ to reflect the newly added point as follows: $\mathbf{c}_i = \mathbf{c}_i + (1/n_i)(\mathbf{x} - \mathbf{c}_i)$. |
| 6 | Repeat steps (3) through (5) until convergence. |

Note that, unlike BKM, OKM traverses the data points in random order, which aims to reduce OKM's sensitivity to the order in which the data points are processed. Studies have shown that for online learning algorithms like OKM, random traversal is preferable to cyclical traversal, which is used in BKM (Bermejo & Cabestany 2002). This is because cyclical presentation may bias an online learning algorithm, especially when dealing with redundant data sets such as image data.

OKM is an instance of the competitive learning paradigm, an unsupervised learning scheme for discovering general features that can be used to classify a set of patterns (Grossberg 1987; Rumelhart & Zipser 1985). In a basic competitive learning algorithm, given a set of randomly distributed units, the units compete for assignment to a given subset of inputs. After the presentation of each input, the closest unit is deemed the

winner and moved closer to the input. Hard competitive learning, also known as winner-take-all learning, consists of algorithms where each input determines the adaptations of a single winning unit. OKM is an instance of hard competitive learning, as *only* the winning unit is moved towards the input presented in each iteration.

Let $\mathbf{x}^{(t)}$ be the input at time $t$ ($t = 1, 2, \ldots$) and $\mathbf{c}^{(t)}$ be the corresponding nearest center (winning unit) with respect to the $\ell_2^2$ dissimilarity (refer to step (2) in the BKM pseudocode). The adaptation equation for $\mathbf{c}^{(t)}$ is given by

$$\boldsymbol{c}^{(t+1)} = \boldsymbol{c}^{(t)} + r(t)(x^{(t)} - \boldsymbol{c}^{(t)}), \tag{1}$$

where $r \in [0,1]$ is the learning rate, which is chosen to satisfy the Robbins–Monro conditions (Robbins & Monro 1951)

$$\lim_{t\to\infty} r(t) = 0, \tag{2a}$$

$$\sum_{t=1}^{\infty} r(t) = \infty, \tag{2b}$$

$$\sum_{t=1}^{\infty} r(t)^2 = \infty. \tag{2c}$$

These conditions ensure that the learning rate decreases at a rate that is fast enough to suppress the noise, but not too fast to avoid premature convergence. By rearranging the adaptation equation above, we obtain

$$\mathbf{c}^{(t+1)} = r(t)\mathbf{x}^{(t)} + (1 - r(t))\mathbf{c}^{(t)}), \tag{3}$$

which shows that new center $\mathbf{c}^{(t+1)}$ is a convex combination of the old center $\mathbf{c}^{(t)}$ and the input data point $\mathbf{x}^{(t)}$. Under mild regularity conditions, OKM converges almost surely to a local minimum (Bottou 1998).

The original OKM algorithm employs a harmonic learning rate $r(t) = 1/t$, which can be generalized using a parameter $p \in (0.5, 1]$, resulting in the hyperharmonic rate $r(t) = t^{-p}$. In theory, the harmonic rate decays too rapidly, while the hyperharmonic rate with $p = 0.5$ gives much better results (Darken & Moody 1990; Wu & Yang 2006; Thompson, Celebi, & Buck 2020).

OKM scans through the input image only once, as opposed to BKM, which scans through the image multiple times. In our earlier work (Thompson, Celebi, & Buck 2020), we showed that OKM obtains very similar results to BKM while being 41 to 300 times faster. Unlike BKM, however, OKM has an element of randomness in it (refer to step (3) in the OKM pseudocode). In our earlier work (Thompson, Celebi, & Buck 2020), we sampled the input image quasirandomly using a low-discrepancy sequence (Bratley 1988) and showed that such a sampling is not only deterministic, but also gives nearly identical results to pseudorandom sampling on average. In this study, we adopt the same quasirandom sampling approach (refer to Thompson, Celebi, & Buck 2020 for details.)

On a historical note, MacQueen (MacQueen 1967) developed the OKM algorithm and coined the term "k-means" in the mid-1960s. However, in time, "k-means" came to refer to the BKM algorithm rather than MacQueen's OKM algorithm. In fact, a vast majority of the clustering literature discusses *only* the BKM algorithm.

**Incremental Online K-Means**

Incremental online k-means (IOKM) is a binary splitting variant of OKM. IOKM is identical to IBKM with two exceptions. First, IOKM uses OKM (rather than BKM) to refine the newly generated centers in each iteration. Second, in IOKM we can safely take $\|\boldsymbol{\epsilon}\|_2 = 0$, while in IBKM $\|\boldsymbol{\epsilon}\|_2$ must be a small positive number. Otherwise, if we set

$\|\boldsymbol{\epsilon}\|_2 = 0$ in IBKM, the left and right children will be identical and the subsequent BKM run will not be able to separate these identical centers, resulting in an empty cluster.

Following our earlier work (Thompson, Celebi, & Buck 2020), we implement OKM as a one-pass algorithm. In other words, we terminate the iterations once the algorithm is presented with $N$ data points. In each iteration, $K$ centers compete to represent the presented data point. Thus, OKM performs on the order of $NK$ computations. IOKM, on the other hand, performs $\log_2 K$ passes over the input image, but the number of centers competing in each pass is different. In pass $t$ ($t \in \{0, \ldots, \log_2 K - 1\}$), $2^{t+1}$ centers compete, resulting in a total of $\sum_{t=0}^{\log_2 K - 1} 2^{t+1} = 2K - 2$ centers competing. Hence, for $K \ll N$, IOKM performs on the order of roughly $2NK$ computations. Consequently, for reasons of fairness, we terminate each call to OKM inside IOKM after $N/2$ iterations rather than $N$.

As mentioned earlier, BKM is very popular in the clustering literature, whereas its incremental version, IBKM, is popular particularly in the vector quantization literature. Despite its significant computational efficiency over BKM, however, OKM does not seem to be as widely used as its batch counterpart. Finally, to the best of our knowledge, IOKM, which is the incremental version of OKM, has not been investigated in the literature.

## Initialization of BKM and OKM

Recall that both BKM and OKM include an initialization step wherein the initial cluster centers are determined. It is well known that initialization is especially important for a batch learning algorithm like BKM (Celebi 2013; Celebi 2015). In this study, we address the initialization problem for BKM and OKM using the maximin algorithm

(Gonzalez 1985). This algorithm begins by taking an arbitrary data point to be the first center $\mathbf{c}_1$. The remaining $(K - 1)$ centers are determined iteratively as follows. For $i \in \{2, \ldots, K\}$, center $\mathbf{c}_i$ is chosen to be the point with the largest minimum-distance to the previously selected $(i - 1)$ centers, *i.e.*,

$$\boldsymbol{c}_i = \arg max_{\mathbf{x} \in X} \, min \, (d(\mathbf{x}, \mathbf{c}_1), \ldots, d(\mathbf{x}, \mathbf{c}_{i-1})), \tag{4}$$

where $d$ is a metric distance (it is common to take $d = \ell_2$). By using $\mathcal{O}(N)$ additional memory, maximin can be implemented in $\mathcal{O}(NK)$ time (see below). We should also mention that Feder and Green (Feder & Greene 1988) described an elaborate implementation of the maximin algorithm with $\mathcal{O}(N \, log \, K)$ time complexity, which is optimal under the algebraic computation tree model. However, this time-optimal maximin formulation is quite complicated and thus primarily of theoretical interest.

The maximin algorithm calls for an arbitrary selection of the first center. Selecting this center uniformly at random from $X$ is customary, but this makes the otherwise deterministic algorithm randomized. In this study, we achieve determinism by taking the first center as the centroid of $X$, which can be computed as $X$ is read from the disk. The pseudocode for maximin is given in Table 4.

Table 4. *Maximin Algorithm*

| Step | Description |
|---|---|
| 1 | Set, $\mathbf{c}_1 = centroid(X)$. Let $d_j$ $(j \in \{1, \ldots, N\})$ denote the distance of $\mathbf{x}_j$ to its nearest center. Set the index of the next center to be found as $i = 2$. Initialize $d_{max} = -\infty$, the maximum distance between any data point and its nearest center. |
| 2 | For each $j \in \{1, \ldots, N\}$, if $d(\mathbf{x}_j, \mathbf{c}_{i-1}) < d_j$, then set $d_j = d(\mathbf{x}_j, \mathbf{c}_{i-1})$. If $d_{max} < d_j$, then update $d_{max}$ and the index of the corresponding point (*i.e.*, $d_{max} = d_j$ and $j^* = j$). |
| 3 | Set $\mathbf{c}_i = \mathbf{x}_{j^*}$ and increment $i$ $(i.e., i = i + 1)$. |
| 4 | Repeat steps (2) and (3) for the remaining $(K - 2)$ centers. |

An interesting and little known property of maximin is that it selects one and only one center from each of the $K$ clusters provided that $X$ contains compact and separated clusters, *i.e.*, each of the possible intra-cluster distances is less than each of the possible inter-cluster distances (Hathaway, Bezdek, & Huband 2006). In other words, maximin is an ideal initializer for k-means for well-clusterable data sets.

# CHAPTER 3 EXPERIMENTAL RESULTS AND DISCUSSION

## Image Set and Parameter Configuration

The proposed IOKM algorithm was tested on eight popular 24-bit test images shown in Fig. 1. Of these images, Baboon (512 × 512), Lenna (512 × 512), and Peppers (512 × 512) are from the USC-SIPI Image Database (http://sipi.usc.edu/database); Motocross (768 × 512) and Parrots (768 × 512) are from the Kodak Lossless True Color Image Suite (http://r0k.us/graphics/kodak/); and Goldhill (720 × 576), Fish (300 × 200), and Pills (800 × 519) are by Lee Crocker, Luiz Velho, and Karel de Gendre, respectively.

The effectiveness of a CQ algorithm was quantified by the Mean Squared Error (MSE) measure given by

$$MSE(I, \tilde{I}) = \frac{1}{HW} \sum_{r=1}^{H} \sum_{c=1}^{W} \left\| I(r,c) - \tilde{I}(r,c) \right\|_2^2, \quad (5)$$

where $I$ and $\tilde{I}$ respectively denote the $H \times W$ original input and quantized output images. MSE represents the average color distortion with respect to $\ell_2^2$.

As mentioned in the Chapter 2 (Incremental Batch K-Means), when a center $\mathbf{c}_k$ is split, the left child inherits its parent's attributes (*i.e.*, $\mathbf{c}_{2k+1} = \mathbf{c}_k$), while the right child becomes a slightly perturbed version of its parent (*i.e.*, $\mathbf{c}_{2k+2} = \mathbf{c}_k + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is an arbitrary vector of small positive $\ell_2$ norm). Let $\boldsymbol{\epsilon} = (\epsilon, \epsilon, \epsilon)$ with $\epsilon \geq 0$. Experiments with $\epsilon \in \{0.0, 0.255, 1.02, 4.08, 16.32\}$ revealed that as long as $\epsilon$ is small, its precise value makes little difference in the MSE obtained. Thus, we used $\epsilon = 0$ for IOKM and $\epsilon = 0.255$ for IBKM (recall that BKM, which is repeatedly called by IBKM, generates empty clusters unless$\|\boldsymbol{\epsilon}\|_2 > 0$).

## Comparison Against Other CQ Algorithms

The proposed IOKM algorithm (and the other k-means variants, *i.e.*, BKM, IBKM, and OKM) were compared to 13 well-known CQ algorithms, namely popularity (POP) (Heckbert 1982), median-cut (MC) (Heckbert 1982), modified popularity (MPOP) (Braudaway 1987), octree (OCT) (Gervautz & Purgathofer 1988), variance-based algorithm (WAN) (Wan, Wong, & Prusinkiewicz 1990), greedy orthogonal bi-partitioning (WU) (Wu 1991), center-cut (CC) (Joy 1993), self-organizing map (SOM) (Dekker 1994), radius-weighted mean-cut (RWM) (Yang & Lin 1996), modified maximin (MMM) (Xiang 1997), split and merge (SAM) (Brun & Mokhtari 2000), variance-cut (VC) (Celebi, Wen, & Hwang 2015), and variance-cut with Lloyd iterations (VCL) (Celebi, Wen, & Hwang 2015).

Among these, SOM, MMM, VCL, BKM, IBKM, OKM, and IOKM are partitional algorithms, whereas the remaining ones are hierarchical algorithms. Brief descriptions of these algorithms (except for IBKM, OKM, and IOKM, which are described in this thesis) can be found in our previous work (Celebi 2009; Celebi 2011; Celebi, Hwang, & Wen 2014; Celebi, Wen, & Hwang 2015).

Table 1 compares the effectiveness of the CQ algorithms quantified by the MSE measure, with the lowest/best values shown in **bold**. Table 2, on the other hand, compares the efficiency of the four k-means based CQ algorithms on three of the test images: Baboon, Lenna, and Peppers. These images were chosen as they have identical dimensions (512 × 512), while the other images have varying dimensions. The efficiency of a CQ algorithm was measured by CPU time in milliseconds (averaged over 10 independent runs). Each of the remaining CQ algorithms was excluded from the

efficiency comparisons for one of two reasons: 1) The algorithm is a hierarchical one that trades effectiveness for efficiency, or 2) The algorithm is a partitional one that is neither effective nor efficient (compared to the four k-means variants). All algorithms were implemented in the C/C++ language and executed on a 1.8GHz Intel Core i7-8665U CPU. The following observations are in order:

- As expected, the partitional algorithms are generally more effective, than the hierarchical ones.
- In general, VC is the most effective hierarchical algorithm.
- Overall, IBKM is the most effective algorithm as it often attains the best MSE, with IOKM usually attaining the second best MSE. IBKM often gives a slightly lower MSE than IOKM, but this comes at a very high computational cost (IOKM is 42 to 385 times faster than IBKM.) The superiority of IOKM can be attributed to the online nature of IOKM and its integrated initialization scheme.
- OKM is more efficient than IOKM for $K < 128$. At $K = 128$, the two algorithms are almost equally efficient. For $K > 128$, IOKM is more efficient than OKM.
- Compared to OKM, IOKM does not require initialization and attains better MSE values, in general. On average, the two algorithms have about the same efficiency.
- The execution time of BKM varies widely among the three images Baboon, Lenna, and Peppers (all of which have the same number of pixels) for a given $K$ value as shown in Table 2. For example, for $K = 64$, clustering Baboon

with BKM took approximately 14.5s, while using the same algorithm to cluster Peppers took approximately 6.5s. On the other hand, for $K = 128$, clustering Baboon with BKM took approximately 27.5s while clustering Peppers took longer to cluster with an approximate time of 34s. The execution time of IBKM is also similarly unpredictable across the images. In contrast to these batch algorithms, the online algorithms exhibit a very steady trend. In other words, for any given K value, OKM and IOKM take nearly constant time for each image.

Figures 2, 4, and 6 show sample quantization results for close-up sections of the Baboon, Peppers, and Pills images, respectively. Figures 3, 5, and 7 show the full-scale error images for the respective images. Given a pair of original and quantized images, the error image was obtained by amplifying the pixelwise normalized $\ell_2$ differences by a factor of four and then negating them for better visualization. It can be seen that the proposed IOKM algorithm performs remarkably well, resulting in clean images with low distortion. Combined with the MSE figures given in Table 1, these error images demonstrate that the proposed algorithm and IBKM produce very similar results.

## CHAPTER 4 CONCLUSIONS AND FUTURE WORK

In this thesis, an effective, efficient, and deterministic CQ algorithm called incremental online k-means (IOKM) was introduced. IOKM is based on Mac-Queen's online k-means (OKM) algorithm, but unlike OKM and many other partitional clustering algorithms, IOKM does not require an explicit center initialization. In addition, unlike OKM, IOKM is deterministic thanks to its quasirandom sampling scheme. This means that one needs to run IOKM *only* once to obtain a high-quality quantization. The performance of IOKM was examined on a diverse set of public test images and compared to those of conventional as well as state-of-the-art CQ algorithms. The results showed that IOKM is competitive with the best algorithm (incremental batch k-means, IBKM) in terms of effectiveness, while being one to two orders of magnitude faster. IOKM is easy to implement and very efficient (requiring about a third of a second to quantize a 512 × 512 image to 256 colors). IOKM is also easy to use because it requires no user-defined parameters other than $K$ (the number of output colors). Apart from the quasirandom sampling part, nothing in the proposed IOKM algorithm makes it specific to image data. Future work includes exploring the applicability of IOKM to higher-dimensional clustering problems.

## REFERENCES


Banerjee, A., Merugu, S., Dhillon, I. S., Ghosh, J., & Lafferty, J. (2005). Clustering with Bregman divergences. Journal of machine learning research, 6(10), 1705–1749.

Bermejo, S., & Cabestany, J. (2002). The effect of finite sample size on on-line k-means. Neurocomputing, 48(1–4), 511–539. doi: 10.1016/S0925-2312(01)00626-9

Bottou, L., & Bengio, Y. (1995). Convergence properties of the k-means algorithms. In Advances in neural information processing systems. 585–592.

Bottou, L. (1998). Online Learning and Stochastic Approximations. In D. Saad (Ed.), On-Line Learning in Neural Networks, Cambridge University Press, 9–42.

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.

Bregman, L. M. (1967). The Relaxation Method of Finding the Common Point of Convex Sets and its Application to the Solution of Problems in Convex Programming. USSR computational mathematics and mathematical physics, 7(3), 200–217. doi: 10.1016/0041-5553(67)90040-7

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. doi: 10.1201/9781420041484

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., 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., 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.(ed.) (2014) Partitional Clustering Algorithms. Springer. doi: 10.1007/978-3-319-09259-1

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

Chang, C. H., Xu, P., Xiao, R., & Srikanthan, T. (2005). New adaptive color quantization method based on self-organizing maps. IEEE transactions on neural networks, 16(1), 237–249. doi: 10.1109/TNN.2004.836543

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

Feder, T., Greene, D. (1988). Optimal Algorithms for Approximate Clustering, in: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 434–444. doi: 10.1145/62212

Forgy, E. (1965) Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classifications. Biometrics, 21, 768–780.

Frackiewicz, M., & Palus, H. (2011). KM and KHM Clustering Techniques for Colour Image Quantisation. In J. M. R. S. Tavares, R. N. Jorge (Eds.), Computational Vision and Medical Image Processing: Recent Trends, 161–174. Springer, Dordrecht. doi: 10.1007/978-94-007-0011-6_9

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, 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

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. Cognitive science, 11(1), 23–63. doi: 10.1016/S0364-0213(87)80025-3

Hathaway R.J., Bezdek J.C., Huband J.M. (2006). Maximin Initialization for Cluster Analysis, in: Proceedings of the 11th Iberoamerican Congress in Pattern Recognition, Springer, 14–26. doi: 10.1007/11892755_2

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

Hu, Y., & Lee, M. G. (2007). K-means-based color palette design scheme with the use of stable flags. Journal of Electronic Imaging, 16(3), 033003. doi: 10.1117/1.2762241

Hu, Y. C., & Su, B. H. (2008). Accelerated k-means clustering algorithm for colour image quantization. The Imaging Science Journal, 56(1), 29–40. doi: 10.1179/174313107X176298

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

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

Orchard, M. T., & Bouman, C. A. (1991). Color quantization of images. IEEE transactions on signal processing, 39(12), 2677–2690. doi: 10.1109/78.107417

Pei, S. C., & Lo, Y. S. (1998). Color image compression and limited display using self-organization Kohonen map. IEEE Transactions on circuits and systems for video technology, 8(2), 191–205. doi: 10.1109/76.664104

Rasti, J., Monadjemi, A., & Vafaei, A. (2011). Color reduction using a multi-stage Kohonen self-organizing map with redundant features. Expert Systems with Applications, 38(10), 13188–13197. doi: 10.1016/j.eswa.2011.04.132

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

Rumelhart, D. E., & Zipser, D. (1985). Feature discovery by competitive learning. Cognitive science, 9(1), 75–112. doi: 10.1016/S0364-0213(85)80010-0

Schaefer, G., & Zhou, H. (2009). Fuzzy clustering for colour reduction in images. Telecommunication Systems, 40(1), 17–25. doi: 10.1007/s11235-008-9143-8

Schaefer, G., Hu, Q., Zhou, H., Peters, J. F., & Hassanien, A. E. (2012). Rough c-means and fuzzy rough c-means for colour quantisation. Fundamenta Informaticae, 119(1), 113–120. doi: 10.3233/FI-2012-729

Thompson, S., Celebi, M. E. & Buck, K. H. (2020) Fast color quantization using MacQueen's k-means algorithm. Journal of Real-Time Image Processing 17 (5), 1609–1624. doi:10.1007/s11554-019-00914-6

Valenzuela, G., Celebi, M. E., & Schaefer, G. (2018, October). Color quantization using coreset sampling. In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2096–210. IEEE. doi: 10.1109/SMC.2018.00361

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, C. H., Lee, C. N., & Hsieh, C. H. (2007). Sample-size adaptive self-organization map for color images quantization. Pattern Recognition Letters, 28(13), 1616–1629. doi: 10.1016/j.patrec.2007.04.005

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 , 3, 126–133. Academic Press. doi: 10.1016/B978-0-08-050754-5.50035-9

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

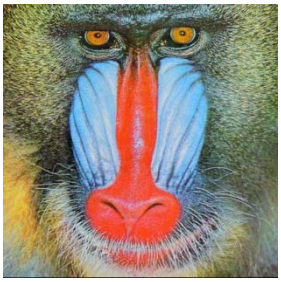
Wu, X., Kumar, V., Quinlan, J. R., et al. (2008). Top 10 Algorithms in Data Mining. Knowledge and Information Systems, 14, 1–37. doi: 10.1007/s10115-007-0114-2

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

Figure 1. *Test Images*



(a) Baboon

(b) Fish

(c) Goldhill

(d) Lenna

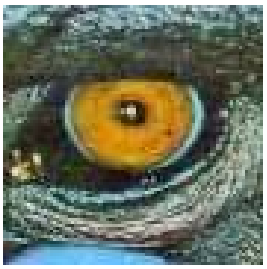
(e) Motocross

(f) Parrots

(g) Peppers

(h) Pills

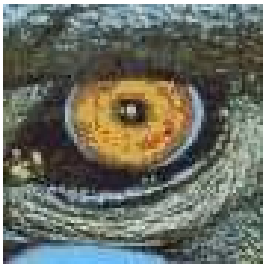
Figure 2. *Baboon output images (K= 32)*



(a) Original

(b) MC output

(c) OCT output

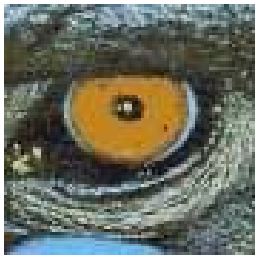


(d) SOM output

(e) VCL output

(f) BKM output

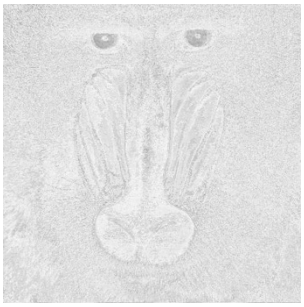
(g) IBKM output



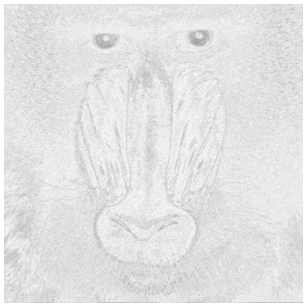
(h) OKM output

(i) IOKM output

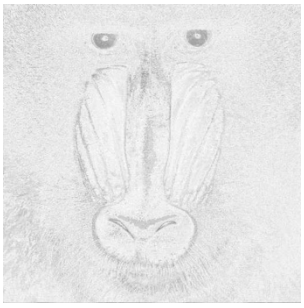
Figure 3. *Baboon error images (K =32)*



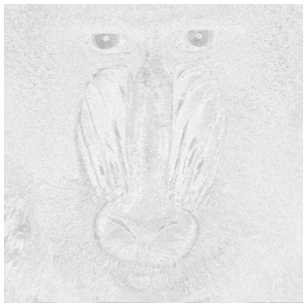
(a) MC error



(b) OCT error



(c) SOM error



(d) VCL error

(e) BKM error

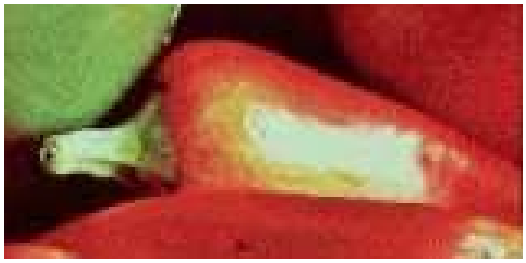
(f) IBKM error

(g) OKM error

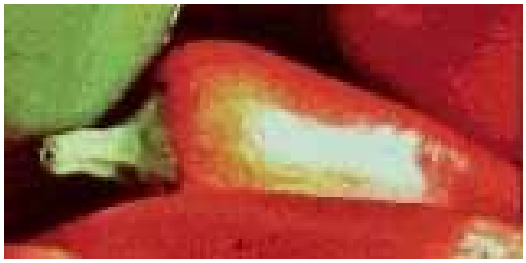
(h) IOKM error

Figure 4. *Peppers output images (K = 64)*

(a) Original

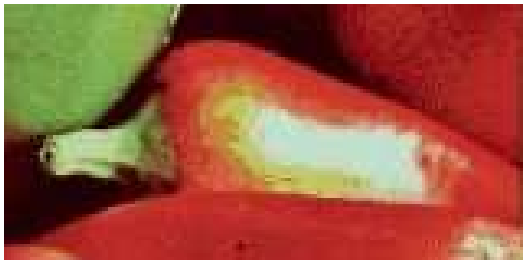


(b) MC output



(c) MMM output

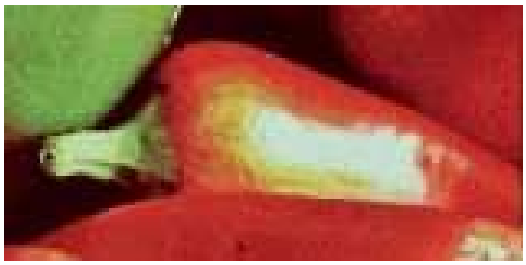
(d) SAM output



(e) VCL output

(f) BKM output

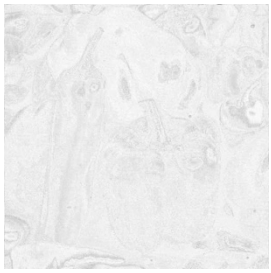
(g) IBKM output



(h) OKM output

(i) IOKM output

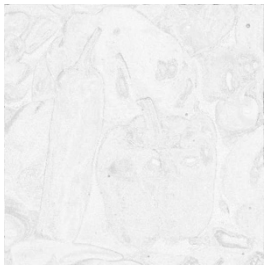
Figure 5. *Peppers error images (K = 64)*



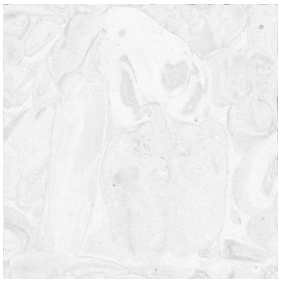
(a) MC error



(b) MMM error

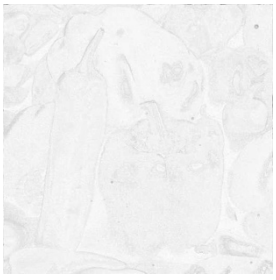


(c) SAM error



(d) VCL error

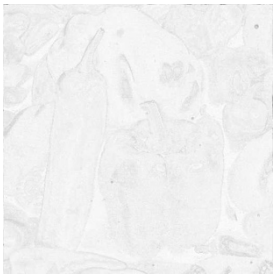
(e) BKM error



(f) IBKM error



(g) OKM error



(h) IOKM error

Figure 6. *Pills output images (K = 128)*

(a) Original

(b) POP output

(c) MPOP output

(d) RWM output

(e) VCL output

(f) BKM output

(g) IBKM output

(h) OKM output

(i) IOKM output

Figure 7. *Pills error images (K = 128)*

(a) POP error

(b) MPOP error



(c) RWM error



(d) VCL error



(e) BKM error



(f) IBKM error



(g) OKM error



(h) IOKM error

Table 5. *MSE comparison of the CQ algorithms*

| Algo | K | | | | K | | | |
|---|---|---|---|---|---|---|---|---|
| | 32 | 64 | 128 | 256 | 32 | 64 | 128 | 256 |
| | Baboon | | | | Fish | | | |
| POP | 1679.5 | 849.5 | 330.7 | 170.4 | 2827.6 | 482.5 | 105.2 | 69.8 |
| MC | 643.0 | 445.6 | 307.4 | 213.0 | 282.3 | 189.4 | 121.2 | 75.9 |
| MPOP | 453.1 | 290.4 | 195.0 | 109.3 | 198.4 | 145.5 | 66.2 | 47.7 |
| OCT | 530.2 | 306.6 | 203.6 | 125.0 | 218.4 | 125.1 | 77.8 | 44.3 |
| WAN | 528.3 | 385.7 | 266.0 | 178.0 | 311.6 | 209.0 | 124.5 | 77.1 |
| WU | 468.3 | 288.3 | 186.5 | 118.6 | 187.6 | 111.6 | 69.0 | 43.8 |
| CC | 473.1 | 299.7 | 202.5 | 144.7 | 189.8 | 127.3 | 82.3 | 56.5 |
| RWM | 459.0 | 301.6 | 188.1 | 120.2 | 176.7 | 109.0 | 68.9 | 44.4 |
| SAM | 464.9 | 293.9 | 188.8 | 119.8 | 198.5 | 120.1 | 74 | 48.5 |
| VC | 450.6 | 273.5 | 179.9 | 117.6 | 168.1 | 106.5 | 67.4 | 43.4 |
| VCL | 425.6 | 264.0 | 173.1 | 115.3 | 169.9 | 102.5 | 65.1 | 43.1 |
| SOM | 433.6 | 268.9 | 163.9 | 108.2 | 180.4 | 114.1 | 60.4 | 45.1 |
| MMM | 510.0 | 368.4 | 230.4 | 147.5 | 223.4 | 144.2 | 81.7 | 53.7 |
| BKM | 374.2 | 234.3 | 149.3 | 95.6 | 142.6 | 90.2 | 57.3 | 34.8 |
| IBKM | **372.6** | **234.0** | **149.2** | **95.3** | **138.1** | **84.6** | **51.2** | **31.8** |
| OKM | 375.7 | 235.2 | 152.2 | 97.7 | 144.5 | 93.1 | 59.0 | 35.9 |
| IOKM | 376.2 | 237.8 | 153.0 | 98.7 | 139.0 | 85.0 | 52.3 | 33.1 |

Table 6. *MSE comparison of the CQ algorithms cont.*

| Algo | K | | | | K | | | |
|---|---|---|---|---|---|---|---|---|
| | 32 | 64 | 128 | 256 | 32 | 64 | 128 | 256 |
| | Goldhill | | | | Lenna | | | |
| POP | 576.7 | 199.3 | 101.8 | 73.1 | 347.2 | 199.5 | 84.5 | 65.3 |
| MC | 293.9 | 188.8 | 132.3 | 86.5 | 214.0 | 146.1 | 112.4 | 80.3 |
| MPOP | 200.2 | 140.7 | 66.7 | 48.6 | 194.5 | 138.9 | 60.0 | 47.8 |
| OCT | 230.3 | 130.3 | 79.0 | 45.7 | 186.7 | 110.0 | 66.0 | 40.6 |
| WAN | 229.0 | 141.2 | 94.5 | 64.4 | 216.5 | 140.8 | 87.6 | 56.7 |
| WU | 196.0 | 114.2 | 71.4 | 45.2 | 158.2 | 99.1 | 61.7 | 39.4 |
| CC | 202.0 | 134.9 | 87.9 | 57.9 | 189.1 | 125.5 | 80.6 | 52.2 |
| RWM | 179.8 | 118.3 | 71.0 | 44.5 | 161.2 | 94.6 | 60.1 | 39.2 |
| SAM | 179.3 | 111.2 | 70.4 | 46.7 | 158.0 | 102.0 | 65.0 | 45.4 |
| VC | 174.8 | 109.5 | 68.3 | 42.4 | 145.6 | 91.7 | 60.7 | 38.9 |
| VCL | 169.3 | 104.3 | 66.2 | 42.0 | 146.3 | 89.2 | 59.2 | 38.6 |
| SOM | 182.1 | 104.2 | 59.5 | 38.4 | 140.2 | 87.4 | 50.5 | 33.9 |
| MMM | 239.9 | 143.1 | 95.4 | 61.0 | 183.3 | 114.2 | 73.5 | 48.5 |
| BKM | 143.8 | **83.0** | **52.0** | 34.2 | 130.8 | 74.7 | 46.8 | 30.3 |
| IBKM | 143.1 | 84.0 | 52.1 | **33.7** | **117.5** | **71.7** | **45.4** | **29.6** |
| OKM | 144.1 | 84.3 | 52.8 | 35.5 | 131.3 | 75.1 | 47.5 | 31.1 |
| IOKM | **141.8** | 83.7 | 52.5 | 34.1 | 119.4 | 72.1 | 46.2 | 30.5 |

Table 7. *MSE comparison of the CQ algorithms cont.*

| Algo | K | | | | K | | | |
|---|---|---|---|---|---|---|---|---|
| | 32 | 64 | 128 | 256 | 32 | 64 | 128 | 256 |
| | Motocross | | | | Parrots | | | |
| POP | 1288.6 | 474.3 | 201.6 | 93.5 | 4086.8 | 371.7 | 180.6 | 104.0 |
| MC | 437.6 | 254.0 | 169.4 | 114.3 | 441.0 | 265.1 | 153.6 | 112.3 |
| MPOP | 287.5 | 177.9 | 84.1 | 53.3 | 379.8 | 212.1 | 104.7 | 59.4 |
| OCT | 300.5 | 158.9 | 96.2 | 54.2 | 342.4 | 191.2 | 111.2 | 63.8 |
| WAN | 445.6 | 292.1 | 168.7 | 92.4 | 376.0 | 233.4 | 153.4 | 92.2 |
| WU | 268.1 | 147.2 | 86.7 | 51.0 | 299.2 | 167.3 | 95.4 | 58.3 |
| CC | 335.1 | 202.0 | 122.6 | 74.9 | 398.8 | 246.5 | 148.7 | 78.9 |
| RWM | 251.4 | 150.1 | 83.7 | 51.0 | 296.5 | 171.0 | 99.8 | 60.6 |
| SAM | 238.1 | 138.5 | 81.8 | 53.5 | 282.4 | 157.5 | 92.4 | 58.8 |
| VC | 253.2 | 144.5 | 79.6 | 48.8 | 290.6 | 166.4 | 98.0 | 58.5 |
| VCL | 240.6 | 131.5 | 77.1 | 47.9 | 263.7 | 157.5 | 96.6 | 57.2 |
| SOM | 301.7 | 134.7 | 70.3 | 44.2 | 279.4 | 151.5 | 82.2 | 47.7 |
| MMM | 407.9 | 276.9 | 138.2 | 85.6 | 352.1 | 194.8 | 128.7 | 68.5 |
| BKM | 197.5 | 115.0 | 68.0 | 42.9 | 230.7 | 129.5 | 73.2 | 44.3 |
| IBKM | **187.9** | **107.8** | 62.3 | **37.0** | 235.0 | 127.7 | **72.4** | **42.3** |
| OKM | 197.3 | 116.4 | 72.4 | 44.9 | 241.0 | 128.5 | 75.4 | 45.0 |
| IOKM | 190.2 | 108.6 | **62.2** | 37.3 | **230.3** | **126.0** | 72.7 | 42.7 |

Table 8. *MSE comparison of the CQ algorithms cont.*

| Algo | K | | | | K | | | |
|---|---|---|---|---|---|---|---|---|
| | 32 | 64 | 128 | 256 | 32 | 64 | 128 | 256 |
| | Peppers | | | | Pills | | | |
| POP | 1389.3 | 367.7 | 218.3 | 129.1 | 788.2 | 222.9 | 124.0 | 85.3 |
| MC | 377.6 | 238.9 | 173.8 | 121.9 | 324.2 | 233.8 | 159.5 | 100.4 |
| MPOP | 338.7 | 204.9 | 112.1 | 69.3 | 277.5 | 175.2 | 88.4 | 55.1 |
| OCT | 317.4 | 193.1 | 113.9 | 68.9 | 281.9 | 159.8 | 99.1 | 56.9 |
| WAN | 348.1 | 225.7 | 157.2 | 106.4 | 294.9 | 197.7 | 133.1 | 87.7 |
| WU | 278.9 | 165.5 | 102.2 | 66.1 | 261.2 | 150.1 | 89.5 | 55.0 |
| CC | 418.4 | 256.8 | 160.7 | 107.9 | 285.9 | 171.7 | 111.9 | 77.4 |
| RWM | 295.6 | 178.8 | 107.1 | 69.2 | 260.4 | 149.7 | 88.8 | 55.6 |
| SAM | 275.7 | 159.2 | 100.8 | 65.9 | 246.2 | 141.2 | 85.0 | 53.7 |
| VC | 294.8 | 169.3 | 108.0 | 69.5 | 234.4 | 146.6 | 90.2 | 54.2 |
| VCL | 261.1 | 160.3 | 103.8 | 68.4 | 229.8 | 141.4 | 85.7 | 53.8 |
| SOM | 270.9 | 160.5 | 89.9 | 69.1 | 226.4 | 137.8 | 72.4 | 46.0 |
| MMM | 341.5 | 213.3 | 136.5 | 85.2 | 276.2 | 174.9 | 117.2 | 75.6 |
| BKM | 248.7 | 148.1 | 87.7 | 55.0 | 198.4 | **111.1** | 66.3 | 41.0 |
| IBKM | **228.9** | **131.8** | **82.7** | **53.1** | 202.4 | 111.7 | **65.5** | **40.1** |
| OKM | 260.8 | 148.9 | 89.3 | 57.3 | 200.1 | 112.7 | 66.9 | 42.0 |
| IOKM | 231.5 | 134.0 | 84.3 | 54.6 | **198.3** | 111.9 | 66.3 | 41.0 |

Table 9. *CPU time comparison of the CQ algorithms*

| Algorithm | *K* | | | |
|---|---|---|---|---|
| | 32 | 64 | 128 | 256 |
| | Baboon | | | |
| BKM | 4432 | 14450 | 27452 | 39083 |
| IBKM | 5573 | 16674 | 73407 | 126005 |
| OKM | 89 | 139 | 220 | 409 |
| IOKM | 132 | 175 | 232 | 328 |
| | Lenna | | | |
| BKM | 4529 | 9690 | 27149 | 33139 |
| IBKM | 5787 | 14229 | 37642 | 72996 |
| OKM | 90 | 140 | 226 | 401 |
| IOKM | 129 | 172 | 230 | 332 |
| | Peppers | | | |
| BKM | 2744 | 6525 | 34149 | 30483 |
| IBKM | 5610 | 16215 | 45464 | 91373 |
| OKM | 92 | 137 | 228 | 403 |
| IOKM | 123 | 169 | 230 | 324 |