Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

The incremental online k-means clustering algorithm and its application to color quantization

Amber Abernathy, M. Emre Celebi [*]

Department of Computer Science and Engineering, University of Central Arkansas, Conway, AR, USA

ARTICLE INFO

Keywords:
Color quantization
Clustering
Batch k-means
Online k-means
Binary splitting
Quasirandom sampling

ABSTRACT

Color quantization is a common image processing operation with various applications in computer graphics, image processing, and computer vision. Color quantization is essentially a large-scale combinatorial optimization problem. Many clustering algorithms, both of hierarchical and partitional types, have been applied to this problem since the 1980s. In general, hierarchical color quantization algorithms are faster, whereas partitional ones produce better results provided that they are initialized properly. In this paper, we propose a novel partitional color quantization algorithm based on a binary splitting formulation of MacQueen's online k-means algorithm. Unlike MacQueen's original algorithm, the proposed algorithm is both deterministic and free of initialization. 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 other words, unlike previously proposed k-means variants, our algorithm addresses both the initialization and acceleration issues of k-means without sacrificing the simplicity of the algorithm. The presented algorithm may be of independent interest as a general-purpose clustering algorithm.

1. Introduction

True-color images have become 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 popular 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. Recent applications of CQ include compression, segmentation, text localization/detection, color analysis, watermarking, non-photorealistic rendering, and content-based retrieval (for specific references, refer to our earlier work Celebi et al., 2015).

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). Pixel mapping can be accomplished using a straightforward linear-time algorithm that maps each input pixel to the nearest palette color. For this reason, most CQ studies (including the present one) 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; Xiang, 2018). 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 et al., 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 on the data (Jain et al., 1999).

Classic hierarchical CQ algorithms include median-cut (Heckbert, 1982), octree (Gervautz & Purgathofer, 1988), Wan et al.'s algorithm (Wan et al., 1990), Wu's algorithm (Wu, 1991), center-cut (Joy & Xiang, 1993), rwm-cut (Yang & Lin, 1996), and pairwise nearest neighbor (Brun & Mokhtari, 2000). More recent hierarchical CQ algorithms include variance-cut (Celebi et al., 2015) and Ueda et al.'s algorithm (Ueda et al., 2017). Partitional algorithms adapted to CQ include maximin (Xiang, 1997), k-means (Celebi, 2009, 2011; Hu & Lee, 2007; Hu & Su, 2008; Huang, 2021; Thompson et al., 2020; Valenzuela et al., 2018), self-organizing maps (Dekker, 1994; Park et al., 2016), fuzzy c-means (Schaefer, 2014; Szilágyi et al., 2016; Wen & Celebi, 2011), k-harmonic means (Frackiewicz & Palus, 2011), rough c-means (Schaefer et al., 2012), and competitive learning (Celebi et al., 2014).

Many of the recent CQ algorithms are based on metaheuristic optimization. Metaheuristics applied to CQ in recent years include

* Corresponding author.
E-mail addresses: aabernathy2@cub.uca.edu (A. Abernathy), ecelebi@uca.edu (M.E. Celebi).

https://doi.org/10.1016/j.eswa.2022.117927
Received 6 November 2021; Received in revised form 24 May 2022; Accepted 18 June 2022
Available online 5 July 2022

differential evolution (Hu et al., 2016), shuffled frog-leaping optimization (Pérez-Delgado, 2019a), artificial bee colony optimization (Huang, 2020), ant colony optimization (Pérez-Delgado, 2021), and flower pollination algorithm (Lei et al., 2020). Some metaheuristics-based CQ algorithms even combine conventional clustering algorithms with metaheuristics (Pérez-Delgado, 2020a; Pérez-Delgado & Gallego, 2019, 2020) or multiple metaheuristics (Pérez-Delgado, 2018, 2019b, 2020b). Metaheuristics-based clustering algorithms are more powerful than conventional clustering algorithms in that they can better optimize nonsmooth and nonconvex objective functions. However, this comes at the price of randomized and complex formulation, an increased number of user-defined parameters, and increased computational requirements. Nevertheless, we should mention that some of the recent metaheuristics-based CQ algorithms (Pérez-Delgado & Gallego, 2019, 2020) have *only* a few user-defined parameters and are reasonably efficient.

Hierarchical CQ algorithms are generally faster, whereas partitional ones often produce better results as long as they are initialized properly. Thanks to the increased computational power in recent years, CQ research has shifted to partitional algorithms. Among these algorithms, most studies have concentrated on k-means (and its soft counterpart, fuzzy c-means) and self-organizing maps. In this paper, we focus on k-means for several reasons. Compared to its alternatives, k-means requires fewer user-defined parameters, is easier to understand and implement, is significantly more efficient (Szilágyi et al., 2016; Wen & Celebi, 2011), and, with suitable initialization, is competitive in terms of the distortion it generates (Celebi, 2009, 2011; Szilágyi et al., 2016; Thompson et al., 2020).

In this paper, we present a novel partitional CQ algorithm based on an online formulation of the k-means algorithm. The main characteristics of the proposed algorithm are the following:

- Unlike standard batch k-means, our algorithm makes *only* one pass over the input image. This formulation renders our algorithm significantly faster than most partitional CQ algorithms proposed earlier.
- Unlike standard batch and online k-means, our algorithm does *not* require an explicit center initialization, which simplifies its implementation.
- Our algorithm produces slightly lower average ℓ_2^2 distortion than standard batch and online k-means, which are considered gold standard algorithms in the CQ literature.
- Our algorithm is deterministic thanks to its quasirandom sampling scheme, which means that *only* a single run is sufficient to achieve good results. By contrast, many partitional CQ algorithms proposed earlier are randomized.
- Thanks to its one-pass formulation and built-in initialization, our algorithm is simpler and easier to implement than its alternatives.

The remainder of the paper is organized as follows. Section 2 first describes three known variants of the k-means clustering algorithm and then the proposed fourth variant. Section 3 presents the experimental results and compares the proposed algorithm to conventional as well as state-of-the-art CQ algorithms. Finally, Section 4 gives the conclusions.

2. K-means and its variants

This section 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.

2.1. 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 et al., 2008). Given a data set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$ and a positive integer $K > 1$, BKM divides $\mathcal{X}$ into K mutually exclusive and exhaustive clusters $\{\mathcal{P}_1, \ldots, \mathcal{P}_K\}$, where each cluster $\mathcal{P}_i$ is represented by a center $\mathbf{c}_i$. The algorithm starts with an arbitrary set of initial centers, customarily chosen uniformly at random from $\mathcal{X}$. 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 $\mathrm{SSE} = \sum_{\mathbf{x}\in\mathcal{X}} d_{\mathrm{SE}}(\mathbf{x}, \{\mathbf{c}_1, \ldots, \mathbf{c}_K\})$, where $d_{\mathrm{SE}}(\mathbf{x}, C)$ denotes the squared Euclidean (ℓ_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. (2005) proved that the optimal center of a cluster is given by the centroid of the cluster *only* for Bregman divergences, a family of nonmetric dissimilarity functions that includes the ℓ_2^2 dissimilarity, squared Mahalanobis dissimilarity, Kullback–Leibler divergence, and Itakura–Saito divergence. In practice, the most popular Bregman divergence is ℓ_2^2.

Let I be the input image in a CQ application. The data set $\mathcal{X}$ 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 below.

```
1. Initialize the cluster centers $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$.
2. Assign each $\mathbf{x} \in \mathcal{X}$ to the nearest center, i.e., $\mathbf{c}_i$ with $i = \arg\min_{k\in\{1,\ldots,K\}} \|\mathbf{x} - \mathbf{c}_k\|_2^2$, 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., $\mathbf{c}_i = (1/n_i)\sum_{\mathbf{x}\in\mathcal{P}_i} \mathbf{x}$, 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 centers stabilize (or, equivalently, when the cluster memberships of the data points 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 et al., 2013).

Although it has a linear time complexity (in N, D, and K), BKM is computationally demanding to due 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, the number of clusters sought, and the initial centers. For this reason, BKM was considered impractical in the early CQ literature (Heckbert, 1982; Wan et al., 1990). More recent studies address this problem by accelerating BKM using a combination of numerical approximations, sampling, weighting, and geometric identities (Celebi, 2009, 2011; Hu & Lee, 2007; Hu & Su, 2008; Huang, 2021; Valenzuela et al., 2018). However, nearly all of these algorithms sacrifice simplicity for computational efficiency. Furthermore, some of them suffer from loss of accuracy (Hu & Lee, 2007; Valenzuela et al., 2018) or convergence guarantee (Hu & Lee, 2007).

2.2. Incremental batch k-means

Incremental batch k-means (IBKM) (Linde et al., 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 below.

```
1. Set $\mathbf{c}_0 = centroid(\mathcal{X})$ and set 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 $(2^{t+1}-1), \dots, (2^{t+2}-2)$ using BKM (the entire data set $\mathcal{X}$ 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 $\mathcal{X}$ (for $K = 1$, this choice of $\mathbf{c}_0$ is clearly optimal). It then adds 2^{t+1} new centers in iteration t ($t \in \{0, \dots, \log_2 K - 1\}$) by 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.*, $\mathbf{c}_{2k+2} = \mathbf{c}_k + \boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon}$ is an arbitrary vector of small positive ℓ_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, we assumed that K is a power of two for simplicity. 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 wherein clustering is employed (Celebi, 2014), IBKM appears to be known primarily in the vector quantization literature from which it originated.

2.3. Online k-means

Online k-means (OKM) (MacQueen, 1967), also known as sequential k-means or MacQueen's k-means, is an online variant of BKM. The two algorithms differ in when and how they update the cluster centers. BKM updates *all* K centers at once after *all* data points are assigned to their respective nearest centers. OKM, on the other hand, updates a *single* 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 & Bengio, 1995). The pseudocode for OKM is given below.

1. Initialize the cluster centers $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$.
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 $\mathcal{X}$ 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 adaptation 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 ($t = 1, 2, \dots$) and $\mathbf{c}^{(t)}$ be the corresponding nearest center (winning unit) with respect to the ℓ_2^2 dissimilarity (refer to step (2) in the BKM pseudocode). The adaptation equation for $\mathbf{c}^{(t)}$ is given by

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

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. Under mild regularity conditions, OKM converges almost surely to a local minimum (Bottou, 1998).

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 $\mathbf{x}^{(t)}$.

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; Thompson et al., 2020; Wu & Yang, 2006).

OKM scans through the input image *only* once, as opposed to BKM, which scans through the image multiple times. [1] In our earlier work (Thompson et al., 2020), we show 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 et al., 2020), we sample the input image quasirandomly using a low-discrepancy sequence (Bratley & Fox, 1988), and demonstrate 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 et al., 2020 for details.)

On a historical note, 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.

2.4. 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 $\|\boldsymbol{\varepsilon}\|_2 = 0$, while in IBKM $\|\boldsymbol{\varepsilon}\|_2$ must be a small positive number. Otherwise, if we set $\|\boldsymbol{\varepsilon}\|_2 = 0$ 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 et al., 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 ($t \in \{0, \dots, \log_2 K - 1\}$), 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

[1] Thompson et al. (2020) show that a single-pass variant of BKM is faster than OKM, but performs very poorly.

Table 1
Comparison of the four k-means variants.

Variant	Mode	Parameters	Initialization	# Passes	Time complexity
BKM	batch	–	explicit	many	$\mathcal{O}(NKD)$
IBKM	batch	ϵ	built-in	many	$\mathcal{O}(NKD)$
OKM	online	p	explicit	one	$\mathcal{O}(NKD)$
IOKM	online	ϵ, p	built-in	one	$\mathcal{O}(NKD)$

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.

Table 1 compares the four k-means variants in terms of their operation mode (batch or online), user-defined parameters, center initialization (explicit or built-in), number of passes over the input image (one or many), and time complexity per pass. Here, an "explicit" initialization refers to the case where the clustering algorithm in question is supplied with appropriate initial centers, whereas a "built-in" initialization refers to the case where the clustering algorithm determines the initial centers by itself. Thus, a clustering algorithm with a built-in initialization can be characterized as an *initialization-free* algorithm. It can be seen that the proposed IOKM algorithm has the most desirable characteristics among the four k-means variants. One could argue that IOKM has more user-defined parameters than its rivals, but, as we demonstrate in Section 3.1, our algorithm is *not* particularly sensitive to the precise value of either of its parameters provided that they are both taken to be small.

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.

2.5. 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 & Kingravi, 2015; Celebi et al., 2013). 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 $i \in \{2, \dots, K\}$, center $\mathbf{c}_i$ is chosen to be the point with the largest minimum-distance to the previously selected $(i-1)$ centers, *i.e.*, $\mathbf{c}_i = \arg\max_{\mathbf{x} \in \mathcal{X}} \min\left(d(\mathbf{x}, \mathbf{c}_1), \dots, d(\mathbf{x}, \mathbf{c}_{i-1})\right)$ 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 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 $\mathcal{X}$ 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 $\mathcal{X}$, which can be computed as $\mathcal{X}$ is read from the disk. The pseudocode for maximin is given below.

1. Set, $\mathbf{c}_1 = \mathit{centroid}(\mathcal{X})$. Let d_j $(j \in \{1, \dots, N\})$ 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.

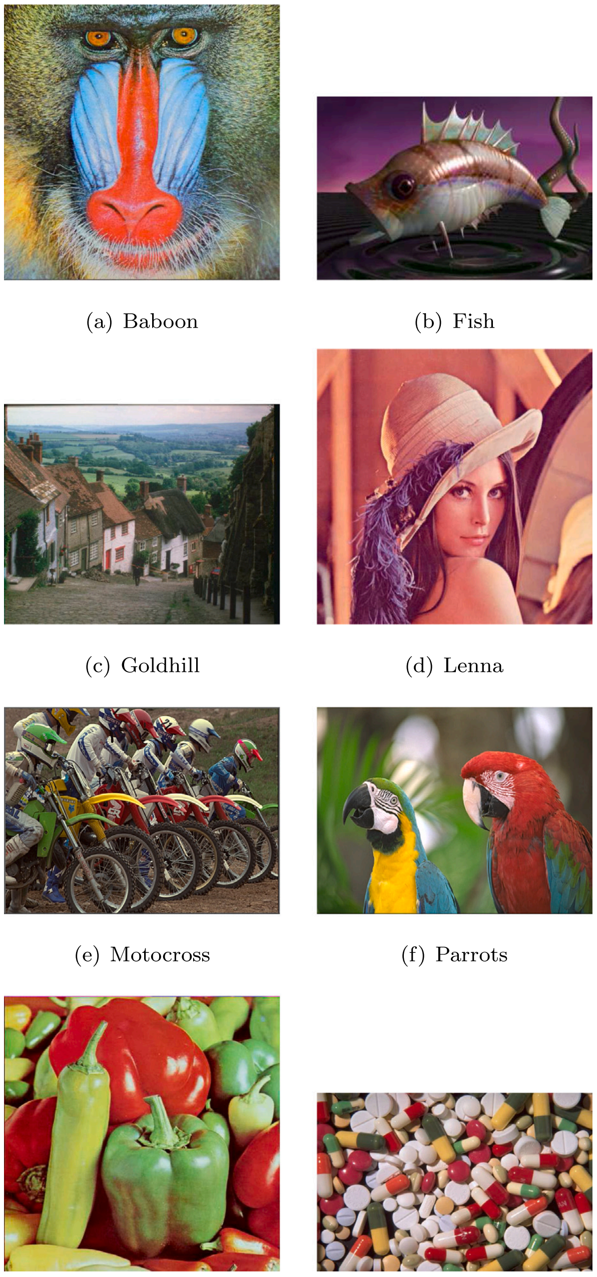

(a) Baboon (b) Fish (c) Goldhill (d) Lenna (e) Motocross (f) Parrots (g) Peppers (h) Pills

Fig. 1. Test images.

2. For each $j \in \{1, \dots, N\}$, 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 *exactly one* center from each of the K clusters provided that $\mathcal{X}$ 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 et al., 2006). In other words, maximin is an ideal initializer for k-means on well-clusterable data sets.

Table 2
Raw MSE results for IBKM.

ε	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Baboon				Fish				Goldhill				Lenna			
0.255	372.6	234.0	149.2	95.3	138.1	84.6	51.2	31.8	143.1	84.0	52.1	33.7	117.5	71.7	45.4	29.6
1.02	374.0	233.5	149.5	95.4	139.4	83.8	50.9	31.6	143.1	84.0	52.1	33.8	117.7	71.4	45.4	29.7
4.08	374.1	233.3	149.7	95.5	136.9	84.6	52.4	32.1	143.1	84.0	52.3	34.0	117.3	71.6	45.5	29.7
16.32	373.1	234.5	149.8	96.3	137.7	85.3	53.6	34.0	146.6	87.2	55.0	36.8	119.4	72.7	46.2	30.7
ε	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Motocross				Parrots				Peppers				Pills			
0.255	187.9	107.8	62.3	37.0	235.0	127.7	72.4	42.3	228.9	131.8	82.7	53.1	202.4	111.7	65.5	40.1
1.02	187.8	106.8	62.3	37.1	235.1	127.5	73.0	42.7	228.8	132.1	82.3	53.0	202.0	111.0	65.5	40.2
4.08	190.0	107.3	62.3	37.1	234.3	125.4	72.0	42.1	228.7	132.0	82.2	53.1	202.0	111.4	65.5	40.4
16.32	189.0	107.9	64.1	39.4	242.4	128.0	73.9	44.2	229.2	132.2	83.0	53.9	198.6	111.2	66.2	41.9

Table 3
Rank MSE results for IBKM.

ε	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Baboon				Fish				Goldhill				Lenna			
0.255	1	3	1	1	3	2	2	2	2	1	1	1	2	3	1	1
1.02	3	2	2	2	4	1	1	1	3	2	2	2	3	1	2	2
4.08	4	1	3	3	1	3	3	3	1	3	3	3	1	2	3	3
16.32	2	4	4	4	2	4	4	4	4	4	4	4	4	4	4	4
ε	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Motocross				Parrots				Peppers				Pills			
0.255	2	3	1	1	2	3	2	2	3	1	3	3	4	4	2	1
1.02	1	1	2	3	3	2	3	3	2	3	2	1	2	1	3	2
4.08	4	2	3	2	1	1	1	1	1	2	1	2	2	3	1	3
16.32	3	4	4	4	4	4	4	4	4	4	4	4	1	2	4	4

Table 4
Raw MSE results for OKM.

p	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Baboon				Fish				Goldhill				Lenna			
0.5	373.5	236.4	151.7	97.9	145.6	93.1	59.0	36.5	144.3	83.8	52.8	35.4	130.4	75.3	47.6	31.2
0.75	376.7	241.6	153.3	99.6	149.9	94.6	61.6	37.7	148.5	87.5	56.0	37.0	130.9	76.3	49.1	32.3
1	387.8	249.8	160.6	105.4	172.6	100.6	63.1	40.1	164.8	95.2	61.2	41.0	137.7	80.7	52.8	34.1
p	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Motocross				Parrots				Peppers				Pills			
0.5	193.6	111.3	71.1	44.7	234.0	128.6	75.7	45.0	258.5	149.9	89.6	57.2	198.6	112.1	67.1	42.2
0.75	212.5	130.9	80.6	49.0	251.8	132.4	81.1	48.3	260.1	150.4	92.0	60.0	200.4	114.3	69.3	43.9
1	245.4	166.0	93.0	55.1	257.9	149.2	86.5	53.1	280.4	167.9	98.7	65.4	210.8	125.2	77.1	49.2

Table 5
Rank MSE results for OKM.

p	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Baboon				Fish				Goldhill				Lenna			
0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0.75	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
p	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Motocross				Parrots				Peppers				Pills			
0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0.75	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Table 6
Raw MSE results for IOKM (Baboon, Fish, Goldhill, Lenna).

ε	p	K				K				K				K			
		32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
		Baboon				Fish				Goldhill				Lenna			
0	0.5	374.3	235.4	151.9	97.4	137.9	84.4	51.8	32.4	142.6	83.3	52.4	33.8	118.4	72.0	46.1	30.1
0	0.75	373.8	237.2	151.8	97.4	138.4	85.1	52.2	32.2	144.8	85.1	52.3	34.1	119.2	72.3	46.1	30.3
0	1	384.6	241.7	156.3	100.0	144.8	87.5	54.9	34.4	148.0	87.2	54.6	36.0	121.4	73.4	47.2	31.2
0.255	0.5	373.5	236.4	152.1	97.3	140.3	84.8	52.0	32.1	142.4	83.8	52.3	33.9	118.1	72.1	46.1	30.1
0.255	0.75	377.5	236.9	152.4	97.3	139.5	85.5	52.7	32.3	142.7	84.4	52.4	33.9	118.6	72.2	46.3	30.2
0.255	1	396.0	246.6	158.8	100.2	147.3	86.8	55.2	33.9	146.2	89.3	54.4	34.7	119.8	73.4	47.8	31.2
1.02	0.5	374.1	236.4	152.1	97.5	138.1	85.8	51.7	32.5	142.5	83.7	52.4	33.7	118.5	71.7	45.9	30.1
1.02	0.75	377.7	237.0	152.6	97.3	138.5	84.3	52.3	32.3	144.8	85.3	53.3	34.3	119.0	72.6	46.0	30.2
1.02	1	391.2	242.6	159.4	101.4	144.4	86.9	53.1	33.4	147.7	86.5	54.4	35.1	121.2	74.1	47.0	31.1
4.08	0.5	374.4	235.8	152.4	97.4	139.7	84.6	52.3	32.7	142.9	83.7	52.5	34.4	118.7	71.8	46.0	30.3
4.08	0.75	374.9	236.3	151.5	98.0	139.5	85.0	52.5	33.3	142.2	85.4	53.1	34.8	118.2	72.4	46.2	30.3
4.08	1	382.8	240.4	160.7	101.2	146.1	89.9	55.0	33.9	146.7	87.7	55.4	35.5	119.7	73.9	47.7	31.7
16.32	0.5	374.7	237.1	152.1	98.4	141.1	86.5	55.6	35.7	145.8	86.6	55.2	36.8	120.4	72.1	47.3	31.6
16.32	0.75	379.4	236.8	155.5	103.6	141.3	89.1	59.7	37.3	147.4	87.1	57.4	39.5	122.2	76.1	48.8	34.4
16.32	1	392.4	253.9	175.5	119.1	148.6	95.6	66.4	41.8	158.8	95.4	65.1	42.7	126.4	80.8	53.3	40.8

Table 7
Raw MSE results for IOKM (Motocross, Parrots, Peppers, Pills).

ε	p	K				K				K				K			
		32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
		Motocross				Parrots				Peppers				Pills			
0	0.5	196.3	106.6	61.5	36.8	234.6	125.2	72.0	42.3	227.3	135.4	83.2	55.2	200.7	110.8	66.0	40.7
0	0.75	187.8	109.1	62.4	37.2	238.9	127.6	73.0	43.4	227.9	135.8	83.5	54.0	201.4	113.1	66.0	41.2
0	1	195.8	112.6	64.5	38.1	257.4	128.1	75.8	46.3	238.7	138.8	86.9	56.4	207.6	118.1	68.3	42.3
0.255	0.5	195.5	106.1	61.6	36.8	231.7	126.2	72.2	42.2	229.4	133.0	83.5	54.6	201.2	111.0	65.7	40.7
0.255	0.75	190.9	107.1	62.7	37.4	240.9	130.1	73.9	42.4	233.6	142.4	83.8	54.0	199.6	114.1	66.3	41.0
0.255	1	195.0	115.8	66.1	38.5	251.1	130.3	75.0	44.7	264.4	144.9	88.8	55.6	204.4	115.3	68.7	42.8
1.02	0.5	193.1	108.5	62.1	37.2	242.5	127.6	72.6	42.2	229.3	135.2	83.1	53.8	199.3	111.3	65.5	40.8
1.02	0.75	195.1	108.1	63.5	37.1	234.7	130.4	73.0	43.0	230.4	133.9	83.5	53.9	205.9	111.8	66.3	40.8
1.02	1	210.2	112.1	65.7	39.1	265.9	131.8	76.9	43.8	233.3	136.6	86.2	55.9	199.3	113.3	68.8	42.3
4.08	0.5	192.1	106.7	62.2	37.1	233.2	126.8	72.3	41.8	237.6	133.0	83.7	54.7	197.2	111.1	65.9	40.9
4.08	0.75	192.9	109.5	62.3	37.2	238.0	135.7	72.8	43.4	230.5	136.5	84.3	55.1	201.8	112.6	66.2	41.7
4.08	1	196.9	111.2	66.6	39.9	241.6	131.6	79.0	45.3	237.1	145.0	86.5	56.0	207.6	116.6	67.9	42.5
16.32	0.5	188.2	108.4	62.5	38.4	234.8	126.2	74.5	44.1	229.9	133.5	86.1	54.8	200.3	112.4	66.0	41.8
16.32	0.75	188.5	114.7	65.8	40.5	240.2	129.2	76.9	50.0	259.5	135.4	85.8	60.6	199.4	112.8	68.9	45.7
16.32	1	198.0	128.3	74.9	48.9	267.7	141.1	82.8	53.6	238.1	153.8	90.0	63.9	213.3	122.2	73.8	53.1

Table 8
Rank MSE results for IOKM (Baboon, Fish, Goldhill, Lenna).

ε	p	K				K				K				K			
		32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
		Baboon				Fish				Goldhill				Lenna			
0	0.5	4	1	3	5	1	2	2	5	4	1	4	2	3	3	4	3
0	0.75	2	10	2	4	3	6	4	2	7	6	1	5	8	7	6	6
0	1	12	12	11	10	12	12	10	12	14	12	11	12	13	10	10	11
0.255	0.5	1	4	5	3	8	4	3	1	2	4	2	4	1	5	5	2
0.255	0.75	8	7	8	1	6	7	8	3	5	5	5	3	5	6	8	5
0.255	1	15	14	12	11	14	10	12	10	10	14	9	8	10	11	13	10
1.02	0.5	3	5	4	7	2	8	1	6	3	3	3	1	4	1	1	1
1.02	0.75	9	8	9	2	4	1	6	4	8	7	8	6	7	9	3	4
1.02	1	13	13	13	13	11	11	9	9	13	9	10	10	12	13	9	9
4.08	0.5	5	2	7	6	7	3	5	7	6	2	6	7	6	2	2	7
4.08	0.75	7	3	1	8	5	5	7	8	1	8	7	9	2	8	7	8
4.08	1	11	11	14	12	13	14	11	11	11	13	13	11	9	12	12	13
16.32	0.5	6	9	6	9	9	9	13	13	9	10	12	13	11	4	11	12
16.32	0.75	10	6	10	14	10	13	14	14	12	11	14	14	14	14	14	14
16.32	1	14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

3. Experimental results and discussion

3.1. 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[2]; Motocross (768×512) and Parrots (768×512) are from the Kodak Lossless True Color Image Suite[3]; and Goldhill (720×576), Fish (300×200), and Pills (800×519) are by Lee Crocker, Luiz Velho, and Karel de Gendre, respectively.

[2] Available at http://sipi.usc.edu/database/.
[3] Available at http://r0k.us/graphics/kodak/.

Table 9
Rank MSE results for IOKM (Motocross, Parrots, Peppers, Pills).

ε	p	K				K				K				K			
		32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
		Motocross				Parrots				Peppers				Pills			
0	0.5	12	2	1	1	3	1	1	4	1	6	2	9	7	1	4	1
0	0.75	1	8	6	6	7	5	6	7	2	8	5	3	9	9	6	7
0	1	11	12	10	9	13	7	11	13	13	11	13	13	13	14	11	11
0.255	0.5	10	1	2	2	1	2	2	3	4	2	3	5	8	2	2	2
0.255	0.75	4	4	8	8	9	9	8	5	9	12	7	4	5	11	8	6
0.255	1	8	14	13	11	12	10	10	11	15	13	14	10	11	12	12	13
1.02	0.5	7	7	3	5	11	6	4	2	3	5	1	1	3	4	1	4
1.02	0.75	9	5	9	3	4	11	7	6	6	4	4	2	12	5	9	3
1.02	1	15	11	11	12	14	13	13	9	8	10	11	11	2	10	13	10
4.08	0.5	5	3	4	4	2	4	3	1	11	1	6	6	1	3	3	5
4.08	0.75	6	9	5	7	6	14	5	8	7	9	8	8	10	7	7	8
4.08	1	13	10	14	13	10	12	14	12	10	14	12	12	14	13	10	12
16.32	0.5	2	6	7	10	5	3	9	10	5	3	10	7	6	6	5	9
16.32	0.75	3	13	12	14	8	8	12	14	14	7	9	14	4	8	14	14
16.32	1	14	15	15	15	15	15	15	15	12	15	15	15	15	15	15	15

Table 10
Mean MSE rank for IBKM, OKM, and IOKM for various parameter combinations (best values are shown in **bold**).

Algo	ε	p	Mean rank
IBKM	**0.255**	–	**2.00**
IBKM	1.02	–	2.09
IBKM	4.08	–	2.19
IBKM	16.32	–	3.69
OKM	–	**0.5**	**1.00**
OKM	–	0.75	2.00
OKM	–	1	3.00
IOKM	**0**	**0.5**	**3.22**
IOKM	0	0.75	5.44
IOKM	0	1	11.53
IOKM	0.255	0.5	3.28
IOKM	0.255	0.75	6.47
IOKM	0.255	1	11.63
IOKM	1.02	0.5	3.75
IOKM	1.02	0.75	6.06
IOKM	1.02	1	10.94
IOKM	4.08	0.5	4.44
IOKM	4.08	0.75	6.81
IOKM	4.08	1	12.06
IOKM	16.32	0.5	8.09
IOKM	16.32	0.75	11.44
IOKM	16.32	1	14.84

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

$$\mathrm{MSE}(I, \tilde{I}) = \frac{1}{HW} \sum_{r=1}^{H} \sum_{c=1}^{W} \left\| I(r,c) - \tilde{I}(r,c) \right\|_2^2, \tag{4}$$

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 ℓ_2^2.

As mentioned in Section 2.2, 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.*, $\mathbf{c}_{2k+2} = \mathbf{c}_k + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is an arbitrary vector of small positive ℓ_2 norm). Without loss of generality, we assume $\boldsymbol{\varepsilon} = (\varepsilon, \varepsilon, \varepsilon)$ with $\varepsilon \geq 0$.

In order to identify suitable parameter values for each k-means variant, we test the following combinations of values:

- BKM (no additional user-defined parameters)
- IBKM ($\varepsilon \in \{0.255, 1.02, 4.08, 16.32\}$)
- OKM ($p \in \{0.5, 0.75, 1\}$)
- IOKM ($\varepsilon \in \{0, 0.255, 1.02, 4.08, 16.32\}$ and $p \in \{0.5, 0.75, 1\}$)

Recall that $\varepsilon = 0$ is disallowed for IBKM, as explained in Section 2.4.

Table 2 gives the raw MSE results for IBKM. In order to facilitate comparisons among the different ε values, we give the MSE ranks for each image and K value combination in Table 3. In this table, lower ranks indicate better ε values, with ranks of 1 and 4 corresponding to the best and worst ε values, respectively. For example, for Baboon and $K = 32$, the parameter values that generate the lowest and highest distortion (as measured by the MSE) are $\varepsilon = 0.255$ and $\varepsilon = 4.08$, respectively. Similarly, Tables 4 and 5 give respectively the raw and rank MSE results for OKM. Finally, Tables 6 and 7 give the raw MSE results for IOKM for the first and second half of the images, respectively; whereas, Tables 8 and 9 give the rank MSE results corresponding to Tables 6 and 7, respectively. In order to aggregate the rank information given in Tables 3, 5, 8, and 9, for each algorithm, we give the mean rank attained by each parameter value combination in Table 10.

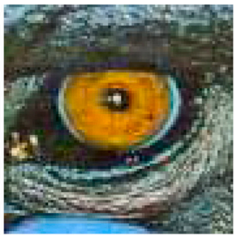
Let us now analyze the parameter sensitivities of IBKM, OKM, and IOKM:

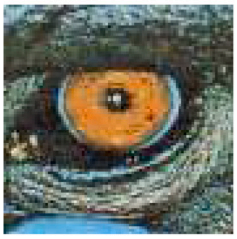
- IBKM: Table 2 shows that IBKM is somewhat insensitive to the value of ε. In other words, for a given image and K value combination, the algorithm attains *more or less* the same MSE regardless of the ε value. Nevertheless, Table 10 shows that $\varepsilon = 0.255$ and $\varepsilon = 16.32$ *generally* give the best and worst results, respectively.
- OKM: Tables 4 and 5 clearly show that $p = 0.5$ and $p = 1$ *consistently* give the best and worst results, respectively. As mentioned in Section 2.3, the disappointing performance of $p = 1$ (*i.e.*, MacQueen's original choice) can be attributed to the fact that this value causes the learning rate to decay too rapidly. These observations are consistent with those reported in our earlier work (Thompson et al., 2020).
- IOKM: Analysis of Tables 6 and 7 (or their respective rank counterparts Tables 8 and 9) is a bit more difficult because of the large (15) number of parameter value combinations. Nevertheless, Table 10 clearly shows that for a given ε value, the mean MSE rank *monotonically* increases with increasing p values, which is consistent with our analysis of OKM. On the other hand, for a given p value, the mean MSE rank *generally* increases with increasing ε values. Hence, smaller ε and p values are generally preferable, with $(\varepsilon, p) = (0, 0.5)$ being the best combination.

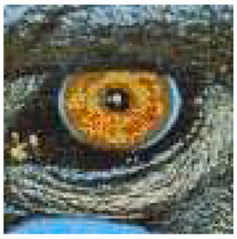
Based on the above analyses, we set the parameters for the four k-means variants as follows: BKM (no additional user-defined parameters), IBKM ($\varepsilon = 0.255$), OKM ($p = 0.5$), and IOKM ($\varepsilon = 0$ and $p = 0.5$).

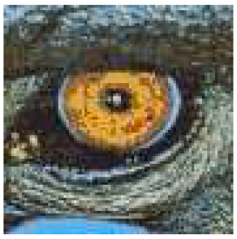
3.2. Comparison against other CQ algorithms

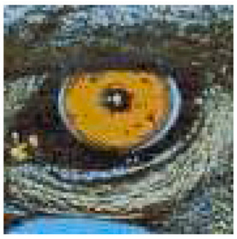
The proposed IOKM algorithm and the other k-means variants (*i.e.*, BKM, IBKM, and OKM) were compared to 13 well-known CQ algorithms, namely popularity (POP) (Heckbert, 1982), median-cut (MC)

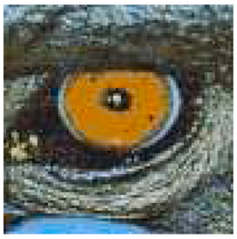

(a) Original

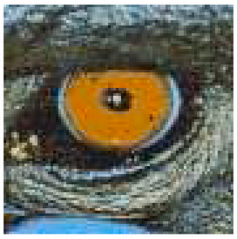

(b) MC output

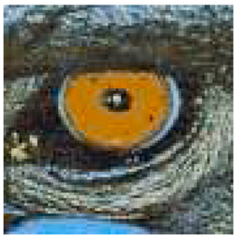

(c) OCT output


(d) SOM output


(e) VCL output


(f) BKM output


(g) IBKM output


(h) OKM output

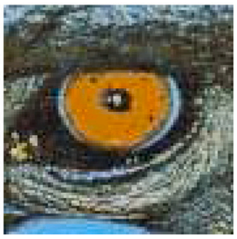
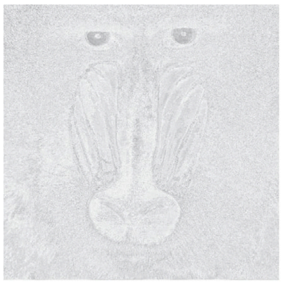
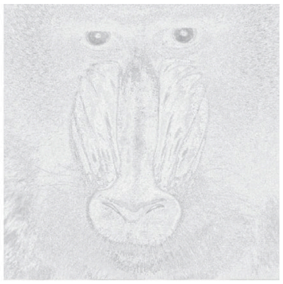

(i) IOKM output

Fig. 2. Baboon output images ($K = 32$).

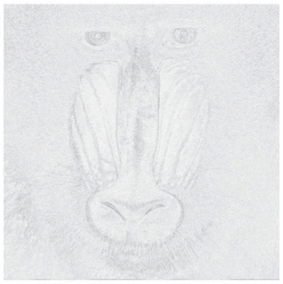

(a) MC error

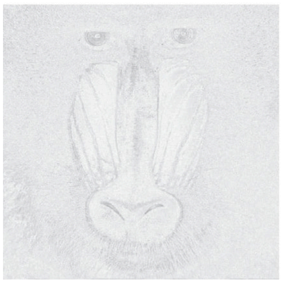

(b) OCT error

(c) SOM error

(d) VCL error

(e) BKM error


(f) IBKM error


(g) OKM error

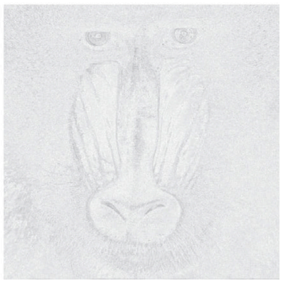
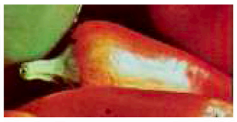

(h) IOKM error

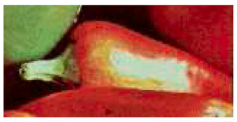
Fig. 3. Baboon error images ($K = 32$).

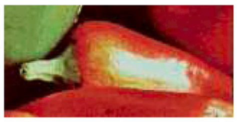
(Heckbert, 1982), modified popularity (MPOP) (Braudaway, 1987), octree (OCT) (Gervautz & Purgathofer, 1988), variance-based algorithm (WAN) (Wan et al., 1990), greedy orthogonal bipartitioning (WU) (Wu, 1991), center-cut (CC) (Joy & Xiang, 1993), self-organizing map (SOM) (Dekker, 1994), radius-weighted mean-cut (RWM) (Yang & Lin, 1996), modified maximin (MMM) (Xiang, 1997), split and merge (SAM) (Brun & Mokhtari, 2000), variance-cut (VC) (Celebi et al., 2015), and variance-cut with Lloyd iterations (VCL) (Celebi et al., 2015).

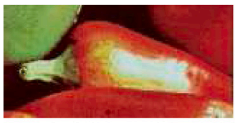
Among these SOM, MMM, VCL, BKM, IBKM, OKM, and IOKM are partitional algorithms, whereas the remaining ones are hierarchical algorithms. Brief descriptions of these algorithms (except IBKM, OKM, and IOKM, which are described in this paper) can be found in our previous work (Celebi, 2009, 2011; Celebi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

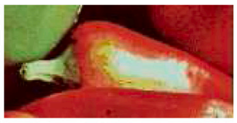
Table 11 compares the effectiveness of the CQ algorithms quantified by the MSE measure, with the lowest/best values shown in **bold**. Table 12, on the other hand, compares the efficiency of the four k-means-based CQ algorithms on three of the test images: Baboon,

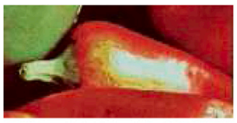

(a) Original

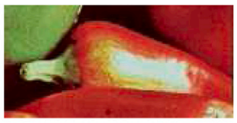

(b) MC output

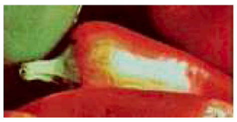

(c) MMM output


(d) SAM output


(e) VCL output


(f) BKM output


(g) IBKM output


(h) OKM output

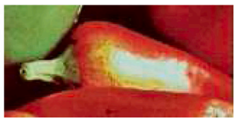
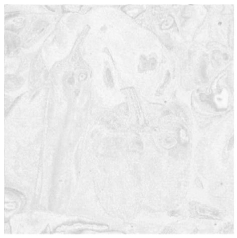
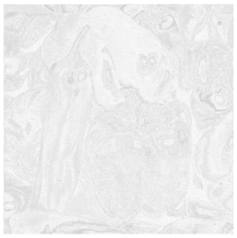
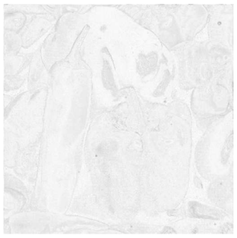

(i) IOKM output

Fig. 4. Peppers output images ($K = 64$).

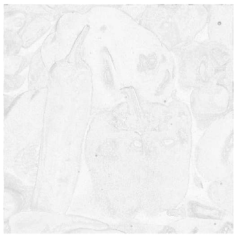

(a) MC error

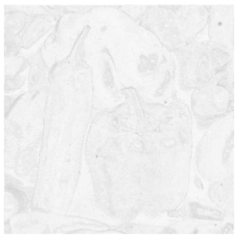

(b) MMM error

(c) SAM error


(d) VCL error

(e) BKM error


(f) IBKM error


(g) OKM error

(h) IOKM error

Fig. 5. Peppers error images ($K = 64$).

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 C/C++, and executed on a 1.8 GHz 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 (IBKM is 42 to 385 times slower than IOKM.) The superiority of IOKM can be attributed to the online nature of IOKM and its built-in 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 an explicit 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 12. For example, for $K = 64$, clustering Baboon with BKM took approximately 14.5 s, while using the same algorithm to cluster Peppers took approximately 6.5 s. On the other hand, for $K = 128$, clustering Baboon with BKM took approximately 27.5 s, while clustering Peppers took longer, 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.

Figs. 2, 4, and 6 show sample quantization results for close-up sections of the Baboon, Peppers, and Pills images, respectively. Figs. 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

(a) Original

(b) POP output

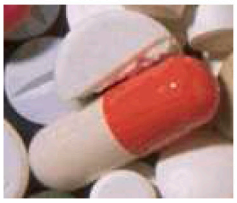
(c) MPOP output

(d) RWM output

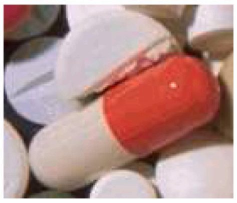
(e) VCL output

(f) BKM output

(g) IBKM output

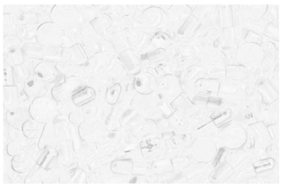


(h) OKM output

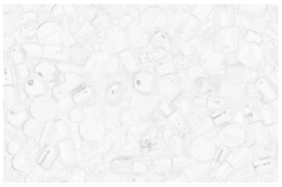


(i) IOKM output

Fig. 6. Pills output images ($K = 128$).



(a) POP error



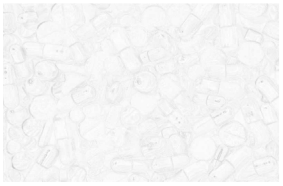
(b) MPOP error

(c) RWM error

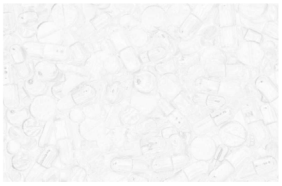
(d) VCL error

(e) BKM error

(f) IBKM error



(g) OKM error



(h) IOKM error

Fig. 7. Pills error images ($K = 128$).

obtained by amplifying the pixelwise normalized ℓ_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 11, these error images demonstrate that the proposed algorithm and IBKM produce very similar results.

4. Conclusions and future work

In this paper, an effective, efficient, and deterministic color quantization algorithm called *incremental online k-means* was introduced. Our algorithm is based on MacQueen's online k-means algorithm, but unlike that algorithm and many other partitional clustering algorithms, ours does *not* require an explicit center initialization. In addition, unlike MacQueen's algorithm, ours is deterministic thanks to its quasirandom sampling scheme. This means that one needs to run our algorithm *only* once to obtain high-quality results. The performance of the proposed algorithm was examined on a diverse set of public test images and compared to those of conventional as well as state-of-the-art color quantization algorithms. The results showed that our algorithm is competitive with the best algorithm (incremental batch k-means) in terms of effectiveness, while being one to two orders of magnitude faster. Our algorithm is easy to implement[4] and very efficient (requiring about a third of a second to quantize a 512 × 512 image to 256 colors). It is also easy to use, requiring no user-defined parameters other than K (the number of output colors). Apart from the quasirandom sampling part, which is introduced to achieve determinism, nothing in the proposed algorithm makes it specific to image data. For example, hierarchical color quantization algorithms are often optimized heavily for integer arithmetic and use lookup tables to accelerate various operations (Celebi et al., 2015). By contrast, our algorithm does *not* exploit the discrete nature of the 24-bit pixel data in any way. This means that, just like the other three k-means variants investigated in this paper, the proposed algorithm can be used to partition any given numerical data set into a predefined number of clusters.

[4] The source code of the proposed algorithm and the other three k-means variants are available at https://github.com/AmberAbernathy.

Table 11
MSE comparison of the CQ algorithms.

Algo	*K*				*K*				*K*				*K*			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Baboon				Fish				Goldhill				Lenna			
POP	1679.5	849.5	330.7	170.4	2827.6	482.5	105.2	69.8	576.7	199.3	101.8	73.1	347.2	199.5	84.5	65.3
MC	643.0	445.6	307.4	213.0	282.3	189.4	121.2	75.9	293.9	188.8	132.3	86.5	214.0	146.1	112.4	80.3
MPOP	453.1	290.4	195.0	109.3	198.4	145.5	66.2	47.7	200.2	140.7	66.7	48.6	194.5	138.9	60.0	47.8
OCT	530.2	306.6	203.6	125.0	218.4	125.1	77.8	44.3	230.3	130.3	79.0	45.7	186.7	110.0	66.0	40.6
WAN	528.3	385.7	266.0	178.0	311.6	209.0	124.5	77.1	229.0	141.2	94.5	64.4	216.5	140.8	87.6	56.7
WU	468.3	288.3	186.5	118.6	187.6	111.6	69.0	43.8	196.0	114.2	71.4	45.2	158.2	99.1	61.7	39.4
CC	473.1	299.7	202.5	144.7	189.8	127.3	82.3	56.5	202.0	134.9	87.9	57.9	189.1	125.5	80.6	52.2
RWM	459.0	301.6	188.1	120.2	176.7	109.0	68.9	44.4	179.8	118.3	71.0	44.5	161.2	94.6	60.1	39.2
SAM	464.9	293.9	188.8	119.8	198.5	120.1	74.0	48.5	179.3	111.2	70.4	46.7	158.0	102.0	65.0	45.4
VC	450.6	273.5	179.9	117.6	168.1	106.5	67.4	43.4	174.8	109.5	68.3	42.4	145.6	91.7	60.7	38.9
VCL	425.6	264.0	173.1	115.3	169.9	102.5	65.1	43.1	169.3	104.3	66.2	42.0	146.3	89.2	59.2	38.6
SOM	433.6	268.9	163.9	108.2	180.4	114.1	60.4	45.1	182.1	104.2	59.5	38.4	140.2	87.4	50.5	33.9
MMM	510.0	368.4	230.4	147.5	223.4	144.2	81.7	53.7	239.9	143.1	95.4	61.0	183.3	114.2	73.5	48.5
BKM	374.2	234.3	149.3	95.6	142.6	90.2	57.3	34.8	143.8	**83.0**	**52.0**	34.2	130.8	74.7	46.8	30.3
IBKM	**372.6**	**234.0**	**149.2**	**95.3**	**138.1**	**84.6**	**51.2**	**31.8**	**143.1**	84.0	52.1	**33.7**	**117.5**	**71.7**	**45.4**	**29.6**
OKM	375.7	235.2	152.2	97.7	144.5	93.1	59.0	35.9	144.1	84.3	52.8	35.5	131.3	75.1	47.5	31.1
IOKM	376.2	237.7	153.7	98.3	141.8	85.6	52.9	33.0	143.5	83.6	52.5	34.2	118.0	72.1	46.3	30.6
Algo	*K*				*K*				*K*				*K*			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
	Motocross				Parrots				Peppers				Pills			
POP	1288.6	474.3	201.6	93.5	4086.8	371.7	180.6	104.0	1389.3	367.7	218.3	129.1	788.2	222.9	124.0	85.3
MC	437.6	254.0	169.4	114.3	441.0	265.1	153.6	112.3	377.6	238.9	173.8	121.9	324.2	233.8	159.5	100.4
MPOP	287.5	177.9	84.1	53.3	379.8	212.1	104.7	59.4	338.7	204.9	112.1	69.3	277.5	175.2	88.4	55.1
OCT	300.5	158.9	96.2	54.2	342.4	191.2	111.2	63.8	317.4	193.1	113.9	68.9	281.9	159.8	99.1	56.9
WAN	445.6	292.1	168.7	92.4	376.0	233.4	153.4	92.2	348.1	225.7	157.2	106.4	294.9	197.7	133.1	87.7
WU	268.1	147.2	86.7	51.0	299.2	167.3	95.4	58.3	278.9	165.5	102.2	66.1	261.2	150.1	89.5	55.0
CC	335.1	202.0	122.6	74.9	398.8	246.5	148.7	78.9	418.4	256.8	160.7	107.9	285.9	171.7	111.9	77.4
RWM	251.4	150.1	83.7	51.0	296.5	171.0	99.8	60.6	295.6	178.8	107.1	69.2	260.4	149.7	88.8	55.6
SAM	238.1	138.5	81.8	53.5	282.4	157.5	92.4	58.8	275.7	159.2	100.8	65.9	246.2	141.2	85.0	53.7
VC	253.2	144.5	79.6	48.8	290.6	166.4	98.0	58.5	294.8	169.3	108.0	69.5	234.4	146.6	90.2	54.2
VCL	240.6	131.5	77.1	47.9	263.7	157.5	96.6	57.2	261.1	160.3	103.8	68.4	229.8	141.4	85.7	53.8
SOM	301.7	134.7	70.3	44.2	279.4	151.5	82.2	47.7	270.9	160.5	89.9	69.1	226.4	137.8	72.4	46.0
MMM	407.9	276.9	138.2	85.6	352.1	194.8	128.7	68.5	341.5	213.3	136.5	85.2	276.2	174.9	117.2	75.6
BKM	197.5	115.0	68.0	42.9	**230.7**	129.5	73.2	44.3	248.7	148.1	87.7	55.0	**198.4**	**111.1**	66.3	41.0
IBKM	**187.9**	**107.8**	**62.3**	**37.0**	235.0	127.7	**72.4**	**42.3**	**228.9**	**131.8**	**82.7**	**53.1**	202.4	111.7	**65.5**	**40.1**
OKM	197.3	116.4	72.4	44.9	241.0	128.5	75.4	45.0	260.8	148.9	89.3	57.3	200.1	112.7	66.9	42.0
IOKM	191.5	108.1	62.9	37.9	241.8	**127.1**	73.4	42.7	230.3	134.0	83.7	54.5	199.1	112.4	66.6	41.1

Table 12
CPU time (in ms.) 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

Future work includes exploring the applicability of the proposed algorithm to higher-dimensional clustering problems. It should be noted that making our algorithm deterministic in a high-dimensional space may be non-trivial, as it is more difficult to cover such a space using a low-discrepancy quasirandom sequence. Therefore, pseudorandom sampling may be unavoidable in high dimensions.

CRediT authorship contribution statement

Amber Abernathy: Software, Formal analysis, Writing – original draft. **M. Emre Celebi:** Conceptualization, Methodology, Software, Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This material is based upon work supported by the National Science Foundation, USA under Award No.OIA-1946391. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

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.
Bottou, L. (1998). Online learning and stochastic approximations. In D. Saad (Ed.), *On-line learning in neural networks* (pp. 9–42). Cambridge University Press.

Bottou, L., & Bengio, Y. (1995). Convergence properties of the K-means algorithms. In *Advances in neural information processing systems 7* (pp. 585–592). MIT Press.

Bratley, P., & Fox, B. L. (1988). Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, *14*(1), 88–100.

Braudaway, G. W. (1987). Procedure for optimum choice of a small number of colors from a large color palette for color imaging. In *Proceedings of the electronic imaging conference* (pp. 71–75).

Brun, L., & Mokhtari, M. (2000). Two high speed color quantization algorithms. In *Proceedings of the 1st international conference on color in graphics and image processing* (pp. 116–121).

Brun, L., & Tremeau, A. (2002). Color quantization. In G. Sharma (Ed.), *Digital color imaging handbook* (pp. 589–638). CRC Press.

Celebi, M. E. (2009). Fast color quantization using weighted sort-means clustering. *Journal of the Optical Society of America A*, *26*(11), 2434–2443.

Celebi, M. E. (2011). Improving the performance of K-means for color quantization. *Image and Vision Computing*, *29*(4), 260–271.

Celebi, M. E. (Ed.), (2014). *Partitional clustering algorithms*. Springer.

Celebi, M. E., Hwang, S., & Wen, Q. (2014). Colour quantisation using the adaptive distributing units algorithm. *Journal of Imaging Science*, *62*(2), 80–91.

Celebi, M. E., & Kingravi, H. A. (2015). Linear, deterministic, and order-invariant initialization methods for the K-means clustering algorithm. In M. E. Celebi (Ed.), *Partitional clustering algorithms* (pp. 79–98). Springer.

Celebi, M. E., Kingravi, H. A., & 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.

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.

Darken, C., & Moody, J. (1990). Fast adaptive K-means clustering: some empirical results. In *Proceedings of the 1990 international joint conference on neural networks, Vol. 2* (pp. 233–238).

Dekker, A. (1994). Kohonen neural networks for optimal colour quantization. *Network. Computation in Neural Systems*, *5*(3), 351–367.

Feder, T., & Greene, D. (1988). Optimal algorithms for approximate clustering. In *Proceedings of the 20th annual ACM symposium on theory of computing* (pp. 434–444).

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, *21*, 768.

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* (pp. 161–174). Springer.

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). Springer.

Gonzalez, T. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, *38*(2–3), 293–306.

Gray, R. (1984). Vector quantization. *IEEE ASSP Magazine*, *1*(2), 4–29.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, *11*(1), 23–63.

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* (pp. 14–26). Springer.

Heckbert, P. (1982). Color image quantization for frame buffer display. *ACM SIGGRAPH Computer Graphics*, *16*(3), 297–307.

Hu, Y. C., & Lee, M. G. (2007). K-means based color palette design scheme with the use of stable flags. *Journal of Electronic Imaging*, *16*(3), Article 033003.

Hu, Y. C., & Su, B. H. (2008). Accelerated K-means clustering algorithm for colour image quantization. *Imaging Science Journal*, *56*(1), 29–40.

Hu, Z., Su, Q., & Xia, X. (2016). Multiobjective image color quantization algorithm based on self-adaptive hybrid differential evolution. *Computational Intelligence and Neuroscience*, *2016*, Article 2450431.

Huang, S. C. (2020). Color image quantization based on the artificial bee colony and accelerated K-means algorithms. *Symmetry*, *12*, 1222.

Huang, S. C. (2021). An efficient palette generation method for color image quantization. *Applied Sciences*, *11*(3), 1043.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323.

Joy, G., & Xiang, Z. (1993). Center-cut for color image quantization. *Visual Computing*, *10*(1), 62–66.

Lei, M., Zhou, Y., & Luo, Q. (2020). Color image quantization using flower pollination algorithm. *Multimedia Tools and Applications*, *79*, 32151–32168.

Linde, Y., Buzo, A., & Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, *28*(1), 84–95.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–136.

MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th berkeley symposium on mathematical statistics and probability, Vol. 1* (pp. 281–297).

Park, H. J., Kim, K. B., & Cha, E. Y. (2016). An effective color quantization method using octree-based self-organizing maps. *Computational Intelligence and Neuroscience*, *2016*, Article 5302957.

Pérez-Delgado, M. L. (2018). Artificial ants and fireflies can perform colour quantisation. *Applied Soft Computing*, *73*, 153–177.

Pérez-Delgado, M. L. (2019a). Color image quantization using the shuffled-frog leaping algorithm. *Engineering Applications of Artificial Intelligence*, *79*, 142–158.

Pérez-Delgado, M. L. (2019b). The color quantization problem solved by swarm-based operations. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *49*, 2482–2514.

Pérez-Delgado, M. L. (2020a). A mixed method with effective color reduction. *Applied Sciences*, *10*(21), 7819.

Pérez-Delgado, M. L. (2020b). Color quantization with particle swarm optimization and artificial ants. *Soft Computing*, *24*, 4545–4573.

Pérez-Delgado, M. L. (2021). Revisiting the iterative ant-tree for color quantization algorithm. *Journal of Visual Communication and Image Representation*, *78*, Article 103180.

Pérez-Delgado, M. L., & Gallego, J. A. R. (2019). A hybrid color quantization algorithm that combines the greedy orthogonal bi-partitioning method with artificial ants. *IEEE Access*, *7*, 128714–128734.

Pérez-Delgado, M. L., & Gallego, J. A. R. (2020). A two-stage method to improve the quality of quantized images. *Journal of Real-Time Image Processing*, *17*(3), 581–605.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*(3), 400–407.

Rumelhart, D., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, *9*(1), 75–112.

Schaefer, G. (2014). Soft computing-based colour quantisation. *EURASIP Journal on Image and Video Processing*, *2014*(1), 1–9.

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.

Szilágyi, L., Dénesi, G., & Enăchescu, C. (2016). Fast color quantization via fuzzy clustering. In *Proceedings of the international conference on neural information processing* (pp. 95–103).

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.

Ueda, Y., Koga, T., Suetake, N., & Uchino, E. (2017). Color quantization method based on principal component analysis and linear discriminant analysis for palette-based image generation. *Optical Review*, *24*(6), 741–756.

Valenzuela, G., Celebi, M. E., & Schaefer, G. (2018). Color quantization using coreset sampling. In *Proceedings of the 2018 IEEE international conference on systems, man, and cybernetics* (pp. 2096–2101).

Wan, S. J., Wong, S. K. M., & Prusinkiewicz (1990). Variance-based color image quantization for frame buffer display. *Color Research and Application*, *15*, 52–58.

Wen, Q., & Celebi, M. E. (2011). Hard vs. Fuzzy C-means clustering for color quantization. *EURASIP Journal on Advances in Signal Processing*, *2011*(1), 118–129.

Wu, X. (1991). Efficient statistical computations for optimal color quantization. In J. Arvo (Ed.), *Graphics gems volume II* (pp. 126–133). Academic Press.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., Zhou, Z. H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37.

Wu, K., & Yang, M. (2006). Alternative learning vector quantization. *Pattern Recognition*, *39*(3), 351–362.

Xiang, Z. (1997). Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics*, *16*(3), 260–276.

Xiang, Z. (2018). Color quantization. In T. F. Gonzalez (Ed.), *Handbook of approximation algorithms and metaheuristics* (2nd ed.). (pp. 691–709). CRC Press.

Yang, C. Y., & Lin, J. C. (1996). RWM-cut for color image quantization. *Computers & Graphics*, *20*(4), 577–588.