Journal of Real-Time Image Processing (2020) 17:1609–1624
https://doi.org/10.1007/s11554-019-00914-6

ORIGINAL RESEARCH PAPER

Fast color quantization using MacQueen's k-means algorithm

Skyler Thompson[1] · **M. Emre Celebi**[1] · **Krizia H. Buck**[1]

Received: 22 May 2019 / Accepted: 6 September 2019 / Published online: 16 October 2019

Abstract
Color quantization (CQ) is an important operation with many applications in computer graphics and image processing and analysis. Clustering algorithms have been extensively applied to this problem. However, despite its popularity as a general purpose clustering algorithm, k-means has not received much attention in the CQ literature because of its high computational requirements and sensitivity to initialization. In this paper, we propose a novel CQ method based on an online k-means formulation due to MacQueen. The proposed method utilizes adaptive and efficient cluster center initialization and quasirandom sampling to attain deterministic, high speed, and high-quality quantization. Experiments on a diverse set of publicly available images demonstrate that the proposed method is significantly faster than the more common batch k-means formulation due to Lloyd while delivering nearly identical results.

Keywords Color quantization · Clustering · MacQueen k-means · Lloyd k-means

1 Introduction

Given an input image, CQ involves the reduction of the number of distinct colors (N') in the image to K ($\ll N'$) with minimum possible distortion. CQ is a challenging problem as most real-world images contain tens of thousands of colors. Recent applications of CQ include compression, segmentation, text localization/detection, color analysis, watermarking, non-photorealistic rendering, and content-based retrieval (for specific references, the reader is referred to our earlier work [8]).

A large number of CQ methods have been developed over the past four decades. These methods can be categorized into two groups: preclustering (hierarchical clustering) methods and postclustering (partitional clustering) methods [7]. The former methods recursively find nested clusters either in a top-down (divisive) or bottom-up (agglomerative) fashion. In contrast, the latter ones find all the clusters simultaneously as a partition of the data and do not impose a hierarchical structure [32]. Compared to preclustering methods, postclustering methods generally produce better results, that is, lower distortion, but are computationally more demanding.

✉ M. Emre Celebi
ecelebi@uca.edu

[1] Department of Computer Science, University of Central Arkansas, Conway, AR, USA

Popular preclustering methods include median-cut [27], octree [22], variance-based method [59], binary splitting [40], greedy orthogonal bipartitioning [61], center-cut [33], RWM-cut [66], and the more recent methods proposed by Celebi et al. [8] and Ueda et al. [56]. On the other hand, postclustering algorithms adapted to CQ include maximin [63], k-means [9, 10, 29, 30, 57], k-harmonic means [21], competitive learning [11, 12], fuzzy c-means [47, 60], rough c-means [50], and self-organizing maps [16, 17, 64].

Recent CQ methods are typically based on metaheuristics or a hybrid of metaheuristics and preclustering/postclustering methods. These methods cast CQ as a global optimization problem, which they then solve by means of a variety of physics- or nature-inspired metaheuristics. Metaheuristics applied to CQ to date include physics-inspired single-solution-based metaheuristics such as simulated annealing [42] as well as nature-inspired population-based metaheuristics such as evolutionary algorithms (genetic algorithms [51], evolution strategies [24], differential evolution [31, 48, 54], etc.) and swarm intelligence algorithms (particle swarm optimization [39], ant colony optimization [43, 44], artificial bee colony optimization [41], artificial fish swarm optimization [20], etc.). These methods are more powerful than preclustering/postclustering methods in that they can optimize nonsmooth, nonconvex objective functions. Unfortunately, these "black-box" methods have several major disadvantages. First they are generally randomized. Second, they

have several parameters that are often difficult to fine tune (initial/final temperature, cooling schedule, population size, crossover/mutation probability, etc.) Third, due to the vast search space in CQ applications, they often require a large number of iterations, which renders them computationally demanding (they can be orders of magnitude slower than k-means). We should, however, mention that some of the recent studies on metaheuristics-based CQ report significantly reduced computational requirements [44].

In this paper, we present an effective and efficient clustering method for CQ. The rest of the paper is organized as follows. Section 2 describes two variants of the k-means clustering algorithm and the proposed k-means based CQ method. Section 3 presents the experimental setup and compares the proposed method to other CQ methods. Finally, Section 4 gives the conclusions.

2 k-means clustering for CQ

In this section, we first describe two common variants of the k-means clustering algorithm, one due to Lloyd and the other due to MacQueen. We then elaborate on the proposed CQ method based on MacQueen's k-means algorithm.

2.1 Lloyd's k-means algorithm

Lloyd's algorithm [35] is perhaps the most common clustering algorithm in scientific and engineering applications [14]. Commonly referred to as (batch) k-means, Lloyd's algorithm starts with a data set[1] $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$ and a positive integer value K, which denotes the desired number of clusters. In the context of CQ, the parameters N, D, and K correspond, respectively, to the number of pixels in the input image, number of color channels (which is typically three), and number of colors desired by the user. The algorithm then assigns each data point $\mathbf{x} \in \mathcal{X}$ to the closest cluster, thereby minimizing the sum of error (SE) given by

$$\mathrm{SE} = \sum_{\mathbf{x}\in\mathcal{X}} d(\mathbf{x}, \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}), \tag{1}$$

where $d(\mathbf{x}, \{\mathbf{c}_1, \ldots, \mathbf{c}_K\})$ denotes the Bregman divergence of $\mathbf{x}$ to the nearest center in $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$, that is

$$d(\mathbf{x}, \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}) = \min_{i\in\{1,\ldots,K\}} d(\mathbf{x}, \mathbf{c}_i).$$

[1] Strictly speaking, in practice, data sets are not implemented as sets, but as sequences, where elements are ordered and duplicates are allowed. This is especially true for image data sets, which are often stored and accessed in raster order.

Bregman divergences are a family of nonmetric dissimilarity functions that include the squared Euclidean distance (ℓ_2^2), squared Mahalanobis distance, Kullback–Leibler divergence, and Itakura–Saito divergence. In practice, the most popular Bregman divergence is the squared Euclidean distance, in which case the SE is called the sum of squared error (SSE). For any Bregman divergence, it can be shown that the optimal center $\mathbf{c}_i$ for cluster $\mathcal{C}_i$ is given by the centroid (or center of mass) of the cluster [1], that is

$$\mathbf{c}_i = \frac{1}{n_i} \sum_{\mathbf{x}\in\mathcal{C}_i} \mathbf{x}, \tag{2}$$

where n_i denotes the cardinality of cluster $\mathcal{C}_i$, that is, the number of points that belong to $\mathcal{C}_i$.

The pseudocode for the k-means algorithm is given below. Starting with a set of K centers, the algorithm alternates between two steps. First, each point is assigned to the nearest cluster. Each cluster center is then recomputed to be the centroid of all points that belong to that cluster. Together, these two steps are referred to as a "Lloyd iteration". These iterations continue until cluster memberships of points no longer change.

1. Let $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ be the initial set of centers.
2. For each $i \in \{1, \ldots, K\}$, set cluster $\mathcal{C}_i$ to be the set of points in $\mathcal{X}$ that are closer in terms of d to $\mathbf{c}_i$ than they are to any other center, that is,

   $$\mathcal{C}_i = \left\{\mathbf{x} \in \mathcal{X} \colon d(\mathbf{x}, \mathbf{c}_i) \le d(\mathbf{x}, \mathbf{c}_{\hat{\imath}}), \text{ for all } \hat{\imath} \ne i\right\}.$$

3. For each $i \in \{1, \ldots, K\}$, set the center $\mathbf{c}_i$ of cluster $\mathcal{C}_i$ to be the centroid of all points in $\mathcal{C}_i$ using Eq. (2).
4. Repeat Lloyd iterations, that is, steps (2) and (3), until convergence.

The most crucial aspect of k-means is step 1, that is, initialization. A common way to determine the initial cluster centers is to select K points uniformly at random from $\mathcal{X}$ and take these as the initial cluster centers. Unfortunately, k-means is relatively sensitive to initialization [13]. Some of the negative effects of improper initialization include empty clusters, slower convergence, and a higher probability of getting stuck at a poor local minimum.

2.2 MacQueen's k-means algorithm

MacQueen [36] proposed an online formulation of the batch k-means algorithm. The two k-means algorithms are similar in the sense that each point is assigned to the cluster represented by the nearest center to that point. The algorithms, however, differ in the way the cluster centers are recomputed. The online algorithm updates the *nearest* center after

the presentation of *each* point, whereas the batch algorithm updates *all* centers after the presentation of the *entire* set of points.

The pseudocode for the online k-means algorithm is given below. It can be seen that, unlike the batch algorithm, this algorithm does not generate the centroidal Voronoi diagram itself, but approximate positions of its generators [19]. This allows the algorithm to achieve $\mathcal{O}(DK)$ per-iteration time complexity, compared to the $\mathcal{O}(NDK)$ time complexity of batch k-means. Therefore, depending on the number of iterations, the online algorithm can be significantly faster than the batch one.

1. Let $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ be the initial set of centers and $n_1 = \cdots = n_K = 1$.
2. Select a random point $\mathbf{x}_r$ from $\mathcal{X}$ and find the nearest center $\mathbf{c}_i$ to this point, that is,

   $$i \leftarrow \arg\min_{\hat{\imath} \in \{1,\ldots,K\}} d(\mathbf{x}_r, \mathbf{c}_{\hat{\imath}}),$$

   where $d(\cdot, \cdot)$ denotes a Bregman divergence as in Lloyd's algorithm.
3. Update the nearest center and the cardinality of the corresponding cluster

   $$\mathbf{c}_i \leftarrow (n_i \mathbf{c}_i + \mathbf{x}_r)/(n_i + 1),$$
   $$n_i \leftarrow n_i + 1.$$

   This ensures that center $\mathbf{c}_i$ now accurately represents the centroid (mean) of all points, including $\mathbf{x}_r$, assigned to the corresponding cluster $\mathcal{C}_i$.
4. Repeat steps (2) and (3) until convergence.

Note that the random presentation of points reduces the sensitivity of the algorithm to the order in which the points are processed. In this regard, random presentation has been shown to be superior to cyclic presentation (that is, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{x}_1, \mathbf{x}_2, \ldots$) [2]. In fact, cyclic presentation is likely to introduce bias into the learning procedure especially in redundant data sets such as image data. Online k-means can be considered as a noisy version of batch k-means [3] and this stochastic noise helps the online algorithm escape from poor local minima more easily.

The online k-means algorithm can also be viewed as an instance of the competitive learning paradigm [25, 46], which is closely related to neural networks. In a basic competitive learning algorithm, we have a randomly distributed set of units that compete for the right to respond to a given subset of inputs [46]. After the presentation of each input, the unit that most closely matches the input (typically in the ℓ_2 sense) is declared as the winner and is moved towards the input. Since only the winner unit is adapted, this kind of learning scheme is termed *hard competitive learning*. Let $\mathbf{x}^{(t)}$ be the input at time t ($t = 1, 2, \ldots$) and $\mathbf{c}^{(t)}$ be the corresponding nearest (winner) unit according to the ℓ_2 distance. The adaptation equation for $\mathbf{c}^{(t)}$ is given by

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

where $\eta \in [0, 1]$ is the learning rate, which is typically a monotonically decreasing function of time. The larger the η value, the more emphasis is given to the new input (this is more obvious in Eq. 5) and hence the faster the learning. Very large values of η, however, may prevent the algorithm from converging. In general, η is chosen to satisfy the Robbins–Monro conditions

$$\lim_{t \to \infty} \eta(t) = 0, \tag{4a}$$

$$\sum_{t=1}^{\infty} \eta(t) = \infty, \tag{4b}$$

$$\sum_{t=1}^{\infty} \eta(t)^2 < \infty. \tag{4c}$$

These conditions ensure that the learning rate decreases fast enough to suppress the noise, but not too fast to avoid premature convergence. Under mild regularity conditions, the algorithm converges almost surely to a local minimum [53, pp. 95–125]. Note that, unlike its batch counterpart, this algorithm does *not* need to process the entire data set to converge. In Sect. 3, we introduce a sampling fraction $f \in (0, 1]$ parameter that can be used to control the proportion of data points that participate in learning.

Rearranging the terms in Eq. (3), we obtain

$$\mathbf{c}^{(t+1)} = \eta(t)\mathbf{x}^{(t)} + (1 - \eta(t))\mathbf{c}^{(t)}, \tag{5}$$

which indicates that the new center $\mathbf{c}^{(t+1)}$ is a convex combination of the old center $\mathbf{c}^{(t)}$ and the input $\mathbf{x}^{(t)}$. In other words, values of η in [0, 1] move $\mathbf{c}$ along the line segment joining $\mathbf{c}$ and $\mathbf{x}$.

The choice $\eta(t) = (1 + t)^{-p}$ with $p \in (0.5, 1]$ guarantees converge [53, p. 109]. The value $p = 1$ gives us the most popular rate in the stochastic approximation literature, $\eta(t) = 1/(1 + t)$, which coincides with MacQueen's choice.[2] This rate gives the fastest asymptotic convergence for $K = 1$ [65]. In practice, however, other rates (e.g., $p = 0.5$ [18, 62]) can lead to faster convergence for finite data sets [53, pp. 95–125], especially when $K > 1$.

Let us elaborate on the computational requirements of Lloyd's and MacQueen's algorithms in terms of their

[2] In MacQueen's algorithm, each unit has its own learning rate, which quantifies the number of times that unit won the competition.

floating-point operation (FLOP) counts. As mentioned earlier, from asymptotic analysis, we know that one iteration of Lloyd's algorithm has the same time complexity, $\mathcal{O}(NDK)$, as one sweep of MacQueen's algorithm (a 'sweep' is one pass through the data set, that is, the successive presentation of N data points). In step (2) of Lloyd's algorithm, for each data point, we compute K ℓ_2^2 distances, find the smallest distance using K comparisons (CMPs), and update the running sum associated with the nearest center along each dimension using D additions (ADDs). It is easy to see that an ℓ_2^2 computation involves D subtractions (SUBs), D multiplications (MULs), and $(D-1)$ ADDs. Hence, assuming that ADD and SUB have the same cost, an ℓ_2^2 computation costs $(2D-1)$ ADDs and D MULs. Totaling for N data points, we see that step (2) of Lloyd's algorithm involves NDK MULs, $N(2DK+D-K)$ ADDs, and NK CMPs. Step (3) involves dividing the running sum associated with each center by its respective cluster size along each dimension, which costs DK divisions (DIVs). Therefore, one iteration of Lloyd's algorithm involves DK DIVs, NDK MULs, $N(2DK+D-K)$ ADDs, and NK CMPs. Now, let us examine the FLOP counts for one sweep of MacQueen's algorithm. In one iteration, we select a data point (SEL), compute K ℓ_2^2 distances, find the smallest distance using K CMPs, determine the learning rate using exponentiation (EXP), and perform adaptation using $2D$ ADDs and D MULs. Therefore, one sweep of MacQueen's algorithm involves N EXPs, N SELs, $ND(K+1)$ MULs, $N(2DK+2D-K)$ ADDs, and NK CMPs. This means that one sweep of MacQueen's algorithm costs an additional N EXPs, N SELs, ND MULs, ND ADDs, minus DK DIVs, over one iteration of Lloyd's algorithm. Since in CQ applications we have $D=3$ and $K \ll N$, the contribution of DIVs is negligible. Therefore, we conclude that, in a CQ application, one sweep of MacQueen's algorithm costs roughly N EXPs, N SELs, $3N$ MULs, and $3N$ ADDs more than one iteration of Lloyd's algorithm. Unfortunately, it is difficult to appreciate the practical implications of this analysis as the actual CPU time requirements of these FLOPs depend on the CPU architecture and the numerical algorithms used for EXP and SEL. In Sect. 3, we compare the two algorithms empirically with respect to CPU time.

2.3 Proposed CQ method

In this section, we describe the proposed CQ method based on MacQueen's k-means algorithm. We would like to devise an effective, efficient, and deterministic CQ method. In other words, we would like our CQ method to produce high-quality results using as little CPU time as possible. We also would like the method to be deterministic so that it would need to be executed only once. We first address the initialization issue and then elaborate upon the sampling issue.

As mentioned earlier, the simplest initialization method involves random selection of the cluster centers. This can be done very efficiently, but the method is not only randomized, but also unreliable. For example, there is no mechanism to avoid choosing centers that are too close to each other. In this study, we explore a well-known alternative to random selection, namely the maximin method [23]. This method selects the first center arbitrarily from the data points and the remaining $(K-1)$ centers are chosen successively as follows. In iteration i ($i \in \{2,\ldots,K\}$), the ith center is chosen to be the point with the greatest minimum distance to the previously selected $(i-1)$ centers, that is, $\mathcal{C}^{(i-1)} = \{\mathbf{c}_1,\ldots,\mathbf{c}_{i-1}\}$. In other words, $\mathbf{c}_i$ is chosen to be $\mathbf{x}_{j^*} \in \mathcal{X}$ with index

$$j^* = \underset{j\in\{1,\ldots,N\}}{\arg\max}\, d\big(\mathbf{x}_j, \mathcal{C}^{(i-1)}\big), \tag{6}$$

where $d(\mathbf{x}_j, \mathcal{C}^{(i-1)})$ is the distance between point $\mathbf{x}_j$ and its nearest center among the previously chosen $(i-1)$ centers, that is

$$d\big(\mathbf{x}_j, \mathcal{C}^{(i-1)}\big) = \min_{\hat{\imath}\in\{1,\ldots,i-1\}} d(\mathbf{x}_j, \mathbf{c}_{\hat{\imath}}). \tag{7}$$

In many cases, the first center $\mathbf{c}_1$ is chosen randomly, but this would render the entire method randomized. A convenient and deterministic alternative is to use the mean data point, that is,

$$\mathbf{c}_1 \leftarrow \frac{1}{N}\sum_{j=1}^{N} \mathbf{x}_j. \tag{8}$$

Now that the first center is chosen deterministically, maximin becomes completely deterministic. Note that the solution to Eq. (6) may not be unique and, thus, for maximin to be deterministic, a deterministic tie-breaking strategy must be used. Perhaps the most common tie-breaking strategy is selecting the smallest index that satisfies Eq. (6), which is computationally very convenient. We should mention that ties are a nuisance that plague *many* clustering algorithms including Lloyd's [52] and MacQueen's algorithms.

A very interesting property of maximin is that the method selects exactly one center from each of the K clusters if the data set is composed of compact and separated clusters, that is, if each of the possible intra-cluster distances is less than each of the possible inter-cluster ones [26]. This can be expressed mathematically as follows:

$$\frac{\min_{i,\hat{\imath}\in\{1,\ldots,K\},\, i\neq\hat{\imath}}\, \min_{\mathbf{x}\in\mathcal{C}_i,\mathbf{x}'\in\mathcal{C}_{\hat{\imath}}} d(\mathbf{x},\mathbf{x}')}{\max_{i\in\{1,\ldots,K\}}\, \max_{\mathbf{x},\mathbf{x}'\in\mathcal{C}_i} d(\mathbf{x},\mathbf{x}')} > 1, \tag{9}$$

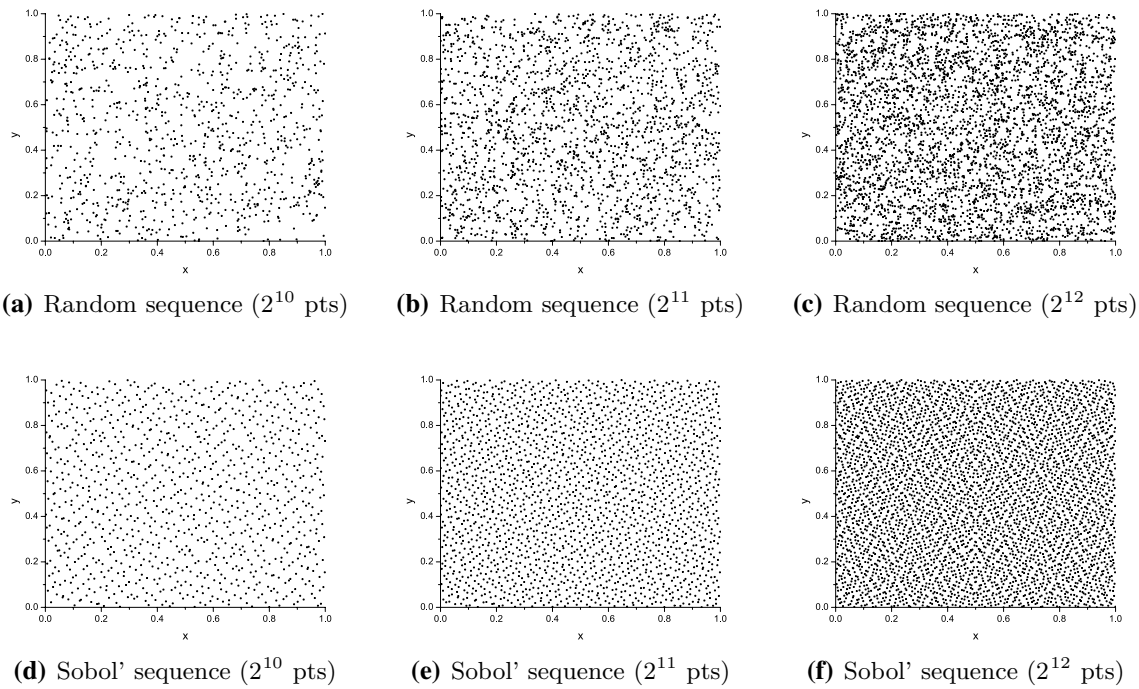
where $d(\cdot,\cdot)$ is a metric.

This greedy initialization method was originally developed as a 2-approximation to the discrete k-center clustering problem, which is defined as follows. Given a set $\mathcal{X}$ of N points in a metric space, find K representative points (centers) such that the maximum distance of any point to its nearest center is minimized. It is NP-hard to approximate this problem to within a factor of less than 2 [28]. Therefore, maximin is the best possible polynomial-time approximation algorithm for this problem.

The pseudocode for the maximin method is given below. It is easy to see that the method has $\mathcal{O}(NDK)$ time complexity.

1. Let $\mathbf{c}_1$ be the first center chosen arbitrarily and $d_1 = \cdots = d_N = \infty$, where d_j denotes the distance of $\mathbf{x}_j$ to its nearest center at any given time. Set the index of the next center to be determined in step 3 as $i \leftarrow 2$.
2. Initialize the maximum distance between any point and its nearest center as $d_{\max} \leftarrow -\infty$ (at the end of this step, the following will hold $d_{\max} = \max(d_1, \ldots, d_N)$). For each $j \in \{1, \ldots, N\}$, if $d(\mathbf{x}_j, \mathbf{c}_{i-1}) < d_j$, then update the distance of $\mathbf{x}_j$ to its nearest center as follows $d_j \leftarrow d(\mathbf{x}_j, \mathbf{c}_{i-1})$. Update $d_{\max}$ and the index of the corresponding point if necessary: if $d_{\max} < d_j$, then $d_{\max} \leftarrow d_j$ and $j^* \leftarrow j$.
3. Set the new center as follows $\mathbf{c}_i \leftarrow \mathbf{x}_{j^*}$ and increment i by one.
4. Repeat steps (2) and (3) a total of $(K - 1)$ times.

Now that we have an efficient and deterministic initialization method at hand, let us turn to the issue of sampling. Due to their online formulation, clustering algorithms such as MacQueen's algorithm are generally more adaptive and thus more likely to escape poor local minima when compared to batch algorithms such as Lloyd's algorithm. The online nature of such clustering algorithms, however, presents two drawbacks. First, these algorithms are order dependent, that is, different presentation orders of the input data points induce different partitions. Second, stochastic selection of the input data points renders these algorithms randomized, that is, each run could potentially generate different clustering results. We avoid these problems by substituting the *pseudorandom* sampling scheme used in MacQueen's algorithm with *quasirandom* sampling. More specifically, we sample the pixel data by means of a quasirandom Sobol' sequence [4]. A quasirandom sequence is a sequence of D-dimensional points that fill $\mathbb{R}^D$ more uniformly than uncorrelated pseudorandom points [45]. This is illustrated in Fig. 1, where the top row shows three pseudorandom sequences with increasing lengths generated by the popular MT19937 generator [37] and the bottom row shows the corresponding quasirandom Sobol' sequences. It is clear that the pseudorandom sequences show clumping, resulting in rather uneven coverage of the sampled region (this would bias the online learning process), whereas the quasirandom sequences produce a significantly more uniform point distribution. It is important to note that, despite



(a) Random sequence (2^{10} pts) **(b)** Random sequence (2^{11} pts) **(c)** Random sequence (2^{12} pts)

(d) Sobol' sequence (2^{10} pts) **(e)** Sobol' sequence (2^{11} pts) **(f)** Sobol' sequence (2^{12} pts)

Fig. 1 Comparison of pseudorandom (**a**–**c**) and quasirandom sampling (**d**–**f**)

their name, quasirandom sequences are indeed completely deterministic. To summarize, the proposed quasirandom sampling scheme imposes a deterministic order on the input data points, thereby rendering MacQueen's algorithm both deterministic and order independent.

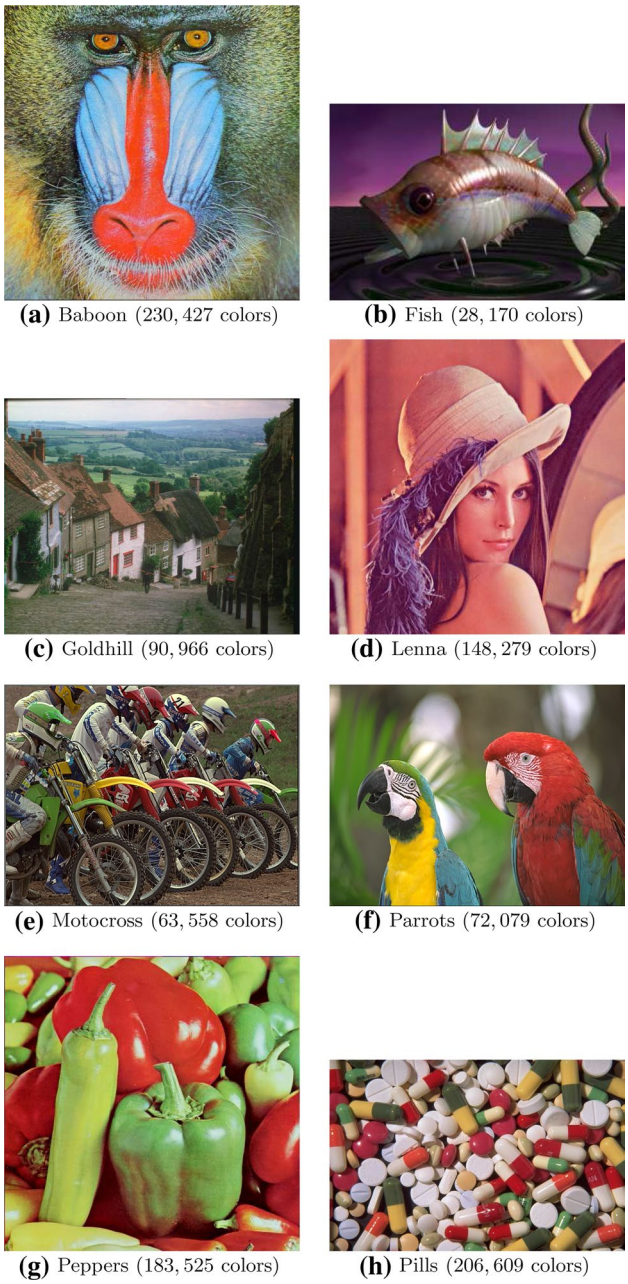
There are various quasirandom sequences including Korobov, Halton, Sobol', Faure, Niederreiter, and Niederreiter–Xing. Among these, Sobol' sequences are often preferred in practice, especially in low dimensions, for several reasons including their favorable uniformity properties, the existence of efficient algorithms to generate them as well as the availability of high-quality implementations of such algorithms (in R, GSL, MATLAB, Boost, NAG Library, etc.) As mentioned earlier, we use Sobol' sequences in the proposed method. Since the theory and implementation of these sequences are involved, the interested reader is referred to the relevant literature [4, 45] for further information.

3 Experimental results and discussion

3.1 Image set and parameter configuration

The proposed CQ method was tested on eight popular 24-bit test images (Fig. 2). Baboon (512×512), Lenna (512×512), and Peppers (512×512) are from the USC-SIPI Image Database; Motocross (768×512) and Parrots (768×512) are from the Kodak Lossless True Color Image Suite; and Goldhill (720×576), Fish (300×200), and Pills (800×519) are by Lee Crocker, Luiz Velho, and Karel de Gendre, respectively.

In addition to K, which is a user-defined parameter in nearly all CQ methods, the proposed method requires two user-defined parameters: exponent $p \in (0.5, 1]$ of the learning rate function and the sampling fraction $f \in (0, 1]$. These parameters control the rate of learning (adaptation) and the proportion of the input pixels that participate in learning (steps 2 and 3 of MacQueen's algorithm, whose pseudocode is given in Sect. 2.2, are each executed Nf times), respectively. To determine the best possible parameter combination for the proposed CQ method, we considered 24 distinct possibilities: $p \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1\} \times f \in \{0.25, 0.5, 0.75, 1\}$. For each of the eight input images and $K \in \{32, 64, 128, 256\}$ value, we quantized the image using the proposed CQ method separately with each of the 24 combinations of (p, f) values and computed the MSE between the input and output images in each case. We then ranked these 24 MSE values with the (p, f) value producing the best, or lowest, MSE value attaining a rank of 1 and the combination producing the worst, or highest, MSE value attaining a rank of 24. Finally, we computed the mean and standard deviation of the MSE ranks across



(a) Baboon (230, 427 colors) **(b)** Fish (28, 170 colors) **(c)** Goldhill (90, 966 colors) **(d)** Lenna (148, 279 colors) **(e)** Motocross (63, 558 colors) **(f)** Parrots (72, 079 colors) **(g)** Peppers (183, 525 colors) **(h)** Pills (206, 609 colors)

Fig. 2 Test images

$8 \times 4 = 32$ combinations of input images and K values. Table 1 gives these rank statistics. It can be seen that $(p, f) = (0.5, 1)$ is the best combination attaining mean and standard deviation ranks of 1.4 and 0.8, respectively, indicating that, this parameter combination almost always generates the lowest distortion. This was not particularly surprising given that within the allowable range of values for p, the fastest learning is given by

Table 1 Rank comparison of the parameter combinations

p	f	Mean	Std
0.5	0.25	9.5	2.8
	0.50	4.8	2.1
	0.75	2.9	1.3
	1.00	**1.4**	**0.8**
0.6	0.25	10.6	2.8
	0.50	6.6	1.7
	0.75	5.4	2.4
	1.00	3.4	2.3
0.7	0.25	14.1	3.1
	0.50	9.8	2.1
	0.75	7.8	2.2
	1.00	7.4	3.3
0.8	0.25	17.5	2.9
	0.50	14.3	1.9
	0.75	13.0	2.5
	1.00	11.6	2.7
0.9	0.25	20.7	1.7
	0.50	17.7	1.8
	0.75	17.3	1.9
	1.00	16.5	2.6
1.0	0.25	23.4	1.2
	0.50	22.0	1.5
	0.75	21.6	1.5
	1.00	20.7	1.7

$p = 0.5$ and, for finite data sets and $K > 1$, $p = 0.5$ may indeed be better than the more common choice $p = 1$, as empirically demonstrated by Darken and Moody [18] and later by Wu and Yang [62]. As for the sampling fraction parameter f, for any given p value, Table 1 shows that mean MSE ranks monotonically decrease with increasing f. This again is not surprising since, in general, the more input pixels are used, the better the learning. In summary, $(p,f) = (0.5, 1)$ seems to be the best parameter combination for the proposed CQ method both from a theoretical and empirical perspective.

3.2 Comparison of CQ methods

Two variants of the proposed method were compared to 15 well-known CQ methods, namely popularity (POP) [27], median-cut (MC) [27], modified popularity (MPOP) [5], octree (OCT) [22], variance-based method (WAN) [59], greedy orthogonal bipartitioning (WU) [61], center-cut (CC) [33], radius-weighted mean-cut (RWM) [66], pairwise clustering (PWC) [58], split and merge (SAM) [6], Cheng and Yang (CY) [15], variance-cut (VC) [8], variance-cut with Lloyd iterations (VCL) [8], self-organizing map (SOM) [17], and modified maxmin (MMM) [63]. Among these, POP, MC, MPOP, OCT, WAN, WU, CC, RWM, PWC, SAM, CY, and VC are preclustering methods, whereas VCL, SOM, and MMM are postclustering methods. The descriptions of these CQ methods can be found in our earlier work [8, 10].

Two variants of each of Lloyd's and MacQueen's algorithms were implemented:

- Batch k-means (BKM) [10]: This method is an adaptation of Lloyd's k-means clustering algorithm (with maximin initialization) to CQ. We examine two variants: one-pass BKM (denoted by BKM_1) and convergent BKM (denoted by BKM). BKM_1 is nothing but the batch version of MacQueen's algorithm where the set of image pixels is presented to the algorithm exactly once (hence the qualifier 'one-pass'). On the other hand, multi-pass convergent BKM performs Lloyd iterations until cluster memberships of points no longer change (hence the qualifier 'convergent').
- MacQueen k-means with pseudorandom sampling (MKM_p): The proposed method with maximin initialization and pseudorandom sampling using the MT19937 (Mersenne Twister) generator. In each iteration, a pseudorandom data point is presented to the algorithm by generating an unbiased pseudorandom integer in the range $[1, N]$ using an efficient algorithm due to Lemire [34], which avoids integer divisions with high probability.
- MacQueen k-means with quasirandom sampling (MKM_q): The proposed method with maximin initialization and quasirandom sampling using a Sobol' sequence. Based on the theoretical and empirical arguments detailed in Sect. 3.1, the adjustable parameters of both MKM variants were set to $(p,f) = (0.5, 1)$. Note that with $f = 1$, MKM variants are in fact one-pass algorithms just like BKM_1.

Table 2 compares the effectiveness (quality) of the CQ methods. The best (lowest) MSE values are shown in bold. For, MKM_q, which is the only randomized CQ method, we give MSE values in the format m_s, where m and s, respectively, denote the mean and standard deviation MSE's over 100 independent runs. The following observations are in order:

Table 2 MSE comparison of the CQ methods

CQ	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
PWC	469.4	308.8	206.7	128.8	201.5	130.9	93.1	69.4	193.8	125.1	88.9	70.9	186.9	108.0	78.8	65.0
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
CY	465.9	280.9	187.3	117.7	193.8	112.5	72.0	44.8	186.3	121.6	72.2	46.4	166.4	97.6	62.5	41.9
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_1	505.0	341.7	218.2	138.2	242.5	139.9	87.3	48.9	250.9	149.3	90.6	61.5	192.9	124.1	72.2	48.1
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**
MKM_p	$375.3_{1.4}$	$236.4_{.6}$	$152.2_{.3}$	$98.2_{.2}$	$147.9_{3.1}$	$93.3_{1.1}$	$59.5_{.5}$	$37.0_{.3}$	$144.4_{.7}$	$84.1_{.5}$	$53.3_{.2}$	$35.6_{.2}$	$131.3_{.5}$	$75.2_{.3}$	$47.7_{.2}$	$31.4_{.1}$
MKM_q	375.6	236.2	152.0	97.6	148.7	92.8	59.2	36.2	144.3	83.3	53.3	35.7	131.4	75.4	47.9	31.3
	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
PWC	243.2	161.2	101.5	78.0	349.4	205.1	125.8	86.0	344.8	183.7	121.1	80.0	283.4	169.3	110.5	75.6
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
CY	248.0	146.6	89.3	53.0	313.2	178.6	106.7	64.5	317.3	186.1	114.1	72.6	237.8	157.9	96.4	58.8
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_1	389.4	237.8	166.2	85.7	363.7	202.1	121.3	71.6	363.1	232.2	138.7	92.8	307.5	188.3	121.5	72.0
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**
MKM_p	$200.2_{4.2}$	$115.9_{1.6}$	$71.9_{.9}$	$45.0_{.4}$	$236.6_{3.6}$	$129.5_{1.2}$	$75.9_{.7}$	$45.2_{.3}$	$257.5_{3.3}$	$148.6_{1.1}$	$89.6_{.4}$	$57.6_{.2}$	$199.7_{1.4}$	$112.3_{.6}$	$67.3_{.4}$	$42.5_{.2}$
MKM_q	**194.3**	116.7	72.7	44.6	241.2	**127.2**	75.9	**44.3**	258.3	148.9	89.4	57.6	199.0	112.0	67.3	42.4

Table 3 CPU time comparison of the k-means based CQ methods ('init': initialization time (msec); 'clust': clustering time (msec); 'cr': clustering time for BKM divided by that for BKM, MKM_p, or MKM_q; 'tr': total time for BKM divided by that for BKM, MKM_p, or MKM_q)

Image	CQ	32				64				128				256			
		init	clust	cr	tr	init	clust	cr	tr	init	clust	cr	tr	init	clust	cr	tr
Baboon	BKM	39	4413	1	1	76	14249	1	1	146	27196	1	1	283	40378	1	1
	MKM_p	37	54	82	49	73	73	195	98	144	111	246	107	283	200	202	84
	MKM_q	37	64	69	44	73	83	171	92	144	122	223	103	283	200	202	84
Lenna	BKM	38	4319	1	1	75	9259	1	1	147	26378	1	1	283	32952	1	1
	MKM_p	36	52	82	49	72	73	127	64	143	109	241	105	283	200	165	69
	MKM_q	36	63	68	44	72	84	111	60	143	122	217	100	283	200	165	69
Peppers	BKM	36	2610	1	1	74	6259	1	1	147	33127	1	1	284	30275	1	1
	MKM_p	37	54	48	29	71	72	87	44	143	110	300	131	284	198	153	63
	MKM_q	36	63	41	27	72	83	76	41	143	122	272	126	284	198	153	63

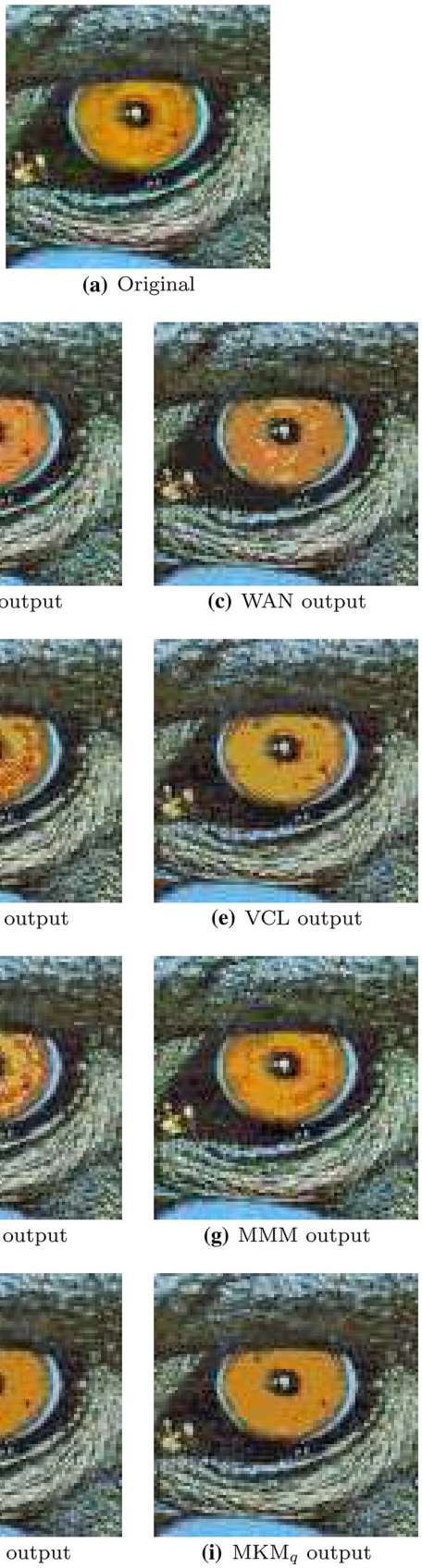
- Unsurprisingly, postclustering methods are generally more effective than the preclustering methods.
- VC and VCL are generally more effective than the other preclustering methods. Note that VCL is not a pure preclustering method as it performs refinement using Lloyd iterations after each split. This hybrid nature of VCL allows it to outperform all preclustering methods.
- MKM_p and MKM_q have similar effectiveness and, in most cases, BKM is slightly more effective than either MKM variant. The MSE differences among the three methods, however, are negligible. For example, whenever BKM is more effective, it outperforms either MKM variant often by only a few units of MSE.
- Despite the fact that both BKM_1 and MKM make a single pass over the image doing roughly the same operations, BKM_1 performs among the worst, whereas MKM performs either the best or a close second. As both methods are initialized in the same manner, the only explanation for the outstanding performance of MKM is its online nature, which helps it learn faster and escape from poor local minima more easily. Note that, as anticipated by the FLOP counts given in Sect. 2.2, BKM_1 is 1.3–2.9 times faster than either MKM variant (data not shown). However, this relative efficiency of BKM_1 in no way makes up for its poor MSE performance.

Table 3 compares the efficiency (speed) of top three CQ methods, namely BKM and the proposed MKM_p and MKM_q methods, on the three most well known images in the CQ literature (Baboon, Lenna, and Peppers, each with resolution 512×512). The remaining CQ methods are either preclustering methods that trade effectiveness for efficiency (with the exception of PWC, these methods take only a few milliseconds of CPU time on a 512×512 image) or postclustering methods (SOM, MMM, and BKM_1) that are neither particularly effective nor particularly efficient. Therefore, we compare the efficiency of only the three most effective CQ methods, which, unsurprisingly, are all postclustering methods. For each K value ($\in \{32, 64, 128, 256\}$) and CQ method, we give four values: initialization time in milliseconds (column 'init'), clustering time in milliseconds (column 'clust'), clustering time for BKM divided by that for BKM, MKM_p, or MKM_q (column 'cr'), and total time for BKM divided by that for BKM, MKM_p, or MKM_q (column 'tr'). As mentioned earlier, all three CQ methods use the same initialization method, namely maximin. Nevertheless, we give initialization time separately for each CQ method so that the reader can judge the computational requirements of initialization in comparison to those of clustering. All methods were implemented in the C language (compiler: gcc v8.2.0, CPU: Intel Core i7-6700 3.40 GHz). The time figures were averaged over 100 independent runs. The following observations are in order:

- Looking at the ratios of total times (column 'tr'), we observe that MKM_p is slightly faster than MKM_q as pseudorandom sampling is slightly more efficient than quasirandom sampling. A few milliseconds of CPU time, however, is a small price to pay for achieving a deterministic sampling scheme. Both MKM variants are significantly faster than BKM (by a factor ranging from 27 to 131). In general, the larger the K value, the more efficient MKM is compared to BKM. On the other hand, if we focus on the ratios of clustering times (column 'cr'), MKM variants become even more advantageous over BKM (by a factor ranging from 41 to 300) as initialization time accounts for a larger percentage of the total time for the MKM variants.

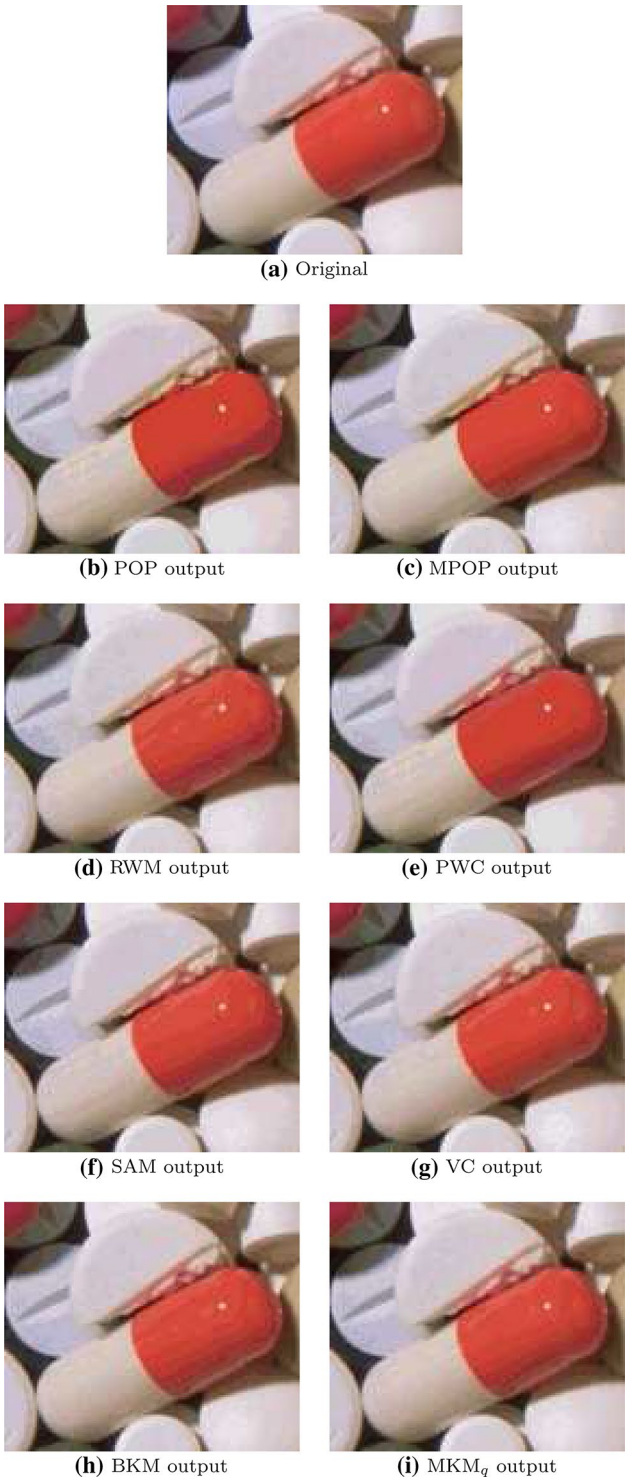
Table 4 Number of iterations for BKM

Image	32	64	128	256
Baboon	186	336	346	269
Lenna	193	221	338	220
Peppers	120	149	428	203



(a) Original
(b) MC output
(c) WAN output
(d) OCT output
(e) VCL output
(f) SOM output
(g) MMM output
(h) BKM output
(i) MKM_q output

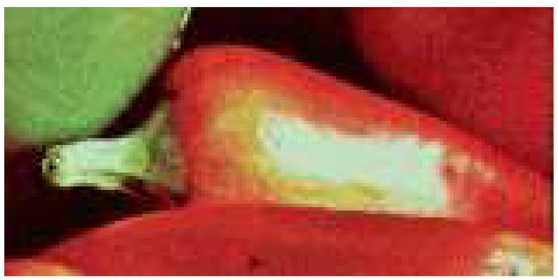
Fig. 3 Baboon output images ($K = 32$)



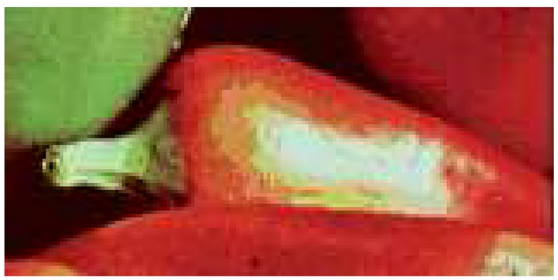
(a) Original
(b) POP output
(c) MPOP output
(d) RWM output
(e) PWC output
(f) SAM output
(g) VC output
(h) BKM output
(i) MKM_q output

Fig. 4 Pills output images ($K = 128$)

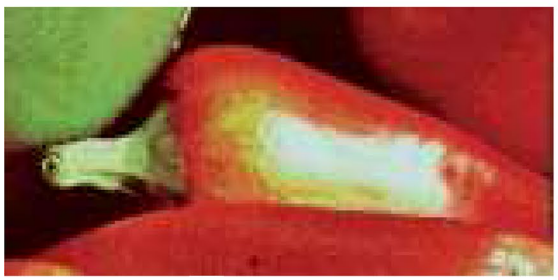
(a) Original



(b) MC output

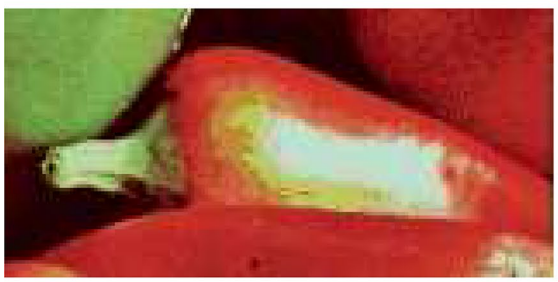


(c) WAN output

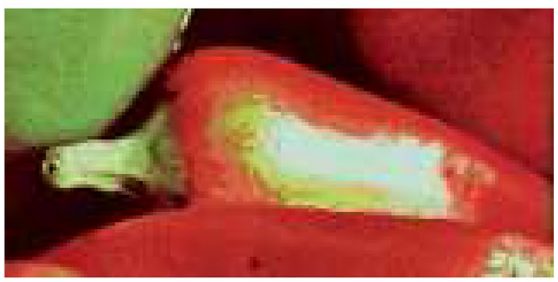


(d) SAM output

(e) CY output



(f) VCL output



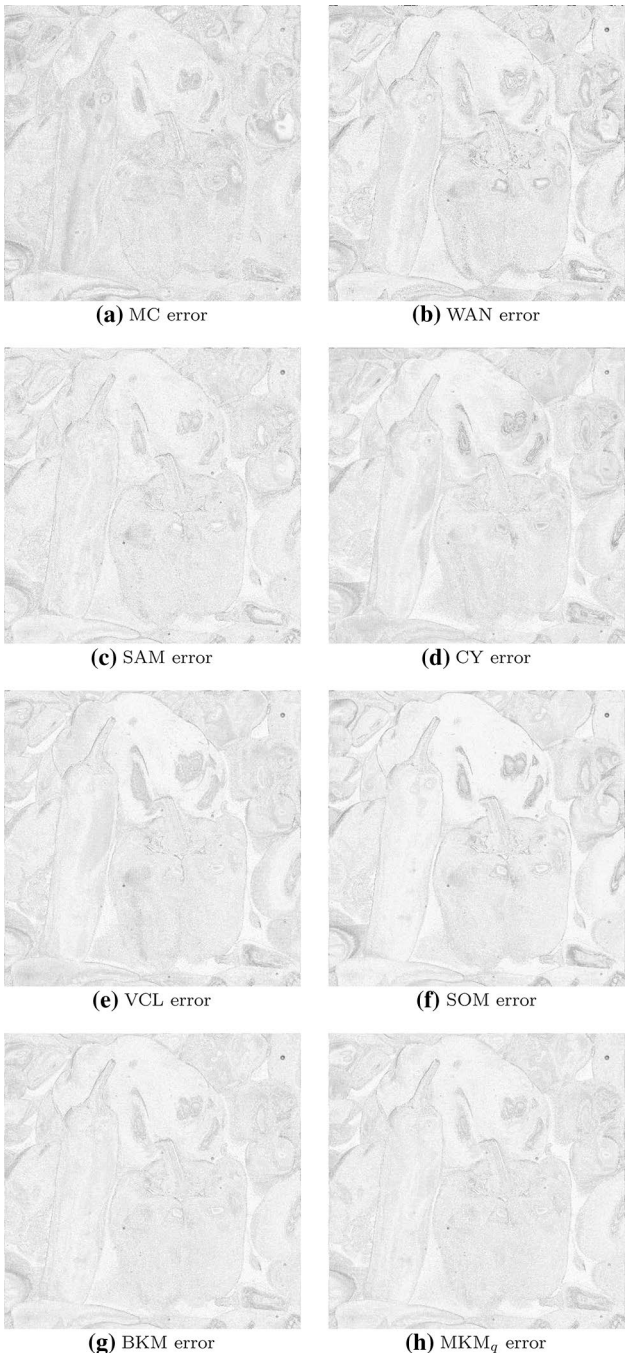
(g) SOM output

(h) BKM output

(i) MKM_q output

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

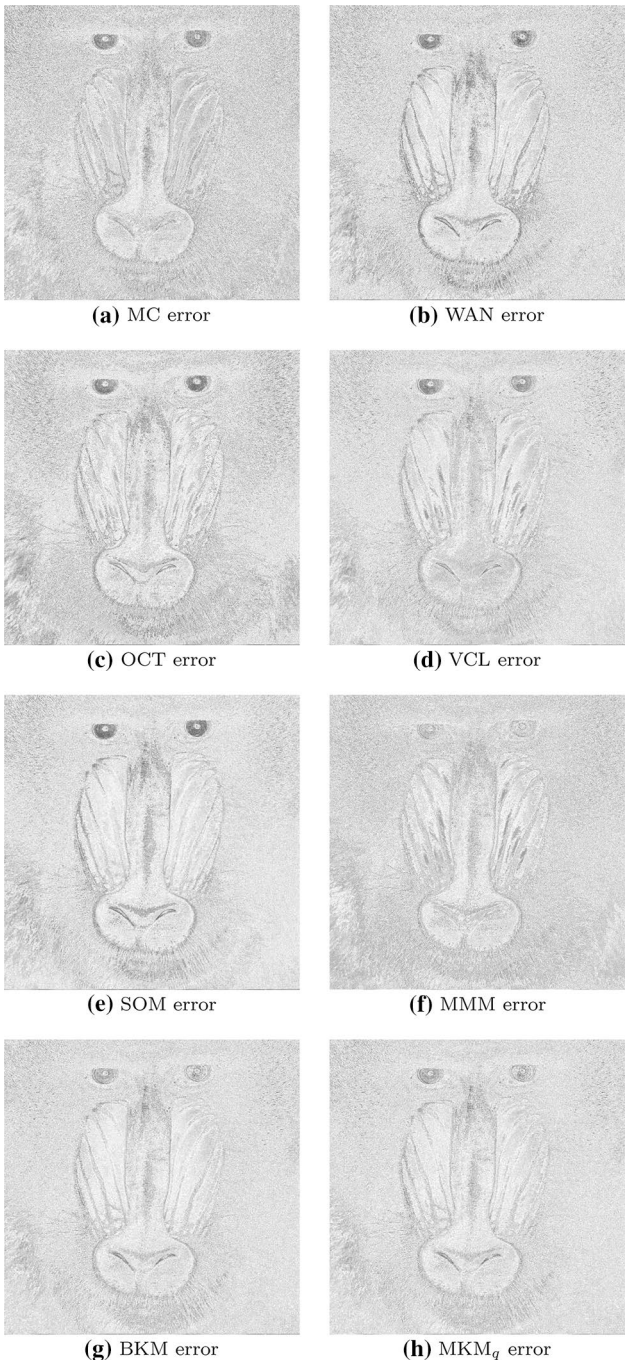
- For BKM, initialization time is negligible compared to clustering time. For MKM, this is not the case; in fact, for $K \geq 64$, initialization takes longer than clustering. The contribution of initialization to the run time of MKM can be minimized using a faster initialization method. For example, preclustering methods such as MC, WAN, and WU are extremely fast, requiring only a few milliseconds on a 512×512 image. However, these methods achieve such impressive efficiency by means of heavily optimized integer arithmetic-based implementations and, consequently, they are applicable for only 8-bit CQ applications, that is, $K \in \{1, \ldots, 256\}$. Our adopted initialization method (maximin), on the other hand, involves no implementation tricks and thus can be used for arbitrary bit-depth CQ, that is, $K \in \{1, \ldots, N'\}$.
- With regard to scalability, the maximin exhibits clearly linear behavior with respect to K. In other words, the initialization time roughly doubles when K is doubled. MKM exhibits sublinear behavior in K, that is, when K is doubled, for example, the clustering time increases by a factor of less than two. Finally, BKM's scalability is unpredictable. In some cases, doubling K increases the clustering time by a factor of more than five (Peppers, $K = 64 \rightarrow 128$), whereas in other cases the clustering time even decreases with increasing K (Peppers, $K = 128 \rightarrow 256$). This unpredictable behavior stems from fact that, for a given image and K value, the number of Lloyd iterations required by BKM cannot be predicted in advance; it depends on various factors including the distribution of colors in the image and the initial centers. Table 4 gives the numbers of Lloyd iterations for BKM. These values explain the unpredictable computational requirements of BKM. For example, clustering Peppers using BKM takes ≈ 33.1 sec. for $K = 128$ and ≈ 30.3 sec. for $K = 256$. From $K = 128$ to $K = 256$, the number of clusters double, but, as Table 4 shows, the number of iterations drops to less than half. This is why BKM is slightly slower for $K = 128$ than it is for $K = 256$.
- For a given K value, the execution time of BKM varies wildly among the three images, all of which have the same number of pixels. For example, for $K = 64$, clustering Baboon with BKM takes ≈ 14.2 s, whereas clustering Peppers with the same method takes ≈ 6.3 s. For $K = 128$, however, the situation is reversed: clustering Peppers takes longer than that of Baboon (≈ 33.1 s vs. ≈ 27.2 s, respectively). As mentioned earlier, these discrepancies can be explained by the differences in the number of iterations (given in Table 4). By contrast, for a particular K value, the execution time of both MKM variants is nearly constant across the three images since both variants are implemented as one-pass algorithms.



(a) MC error **(b)** WAN error **(c)** SAM error **(d)** CY error **(e)** VCL error **(f)** SOM error **(g)** BKM error **(h)** MKM_q error

Fig. 6 Peppers error images ($K = 64$)

Figures 3, 5 and 4 show sample quantization results for close-up parts of the Baboon, Peppers, and Pills images, respectively. Figures 7, 6 and 8 show the full-scale error images for the respective images. The error image for a particular CQ method was obtained by taking the pixelwise absolute difference between the original and quantized images. For



(a) MC error **(b)** WAN error **(c)** OCT error **(d)** VCL error **(e)** SOM error **(f)** MMM error **(g)** BKM error **(h)** MKM_q error

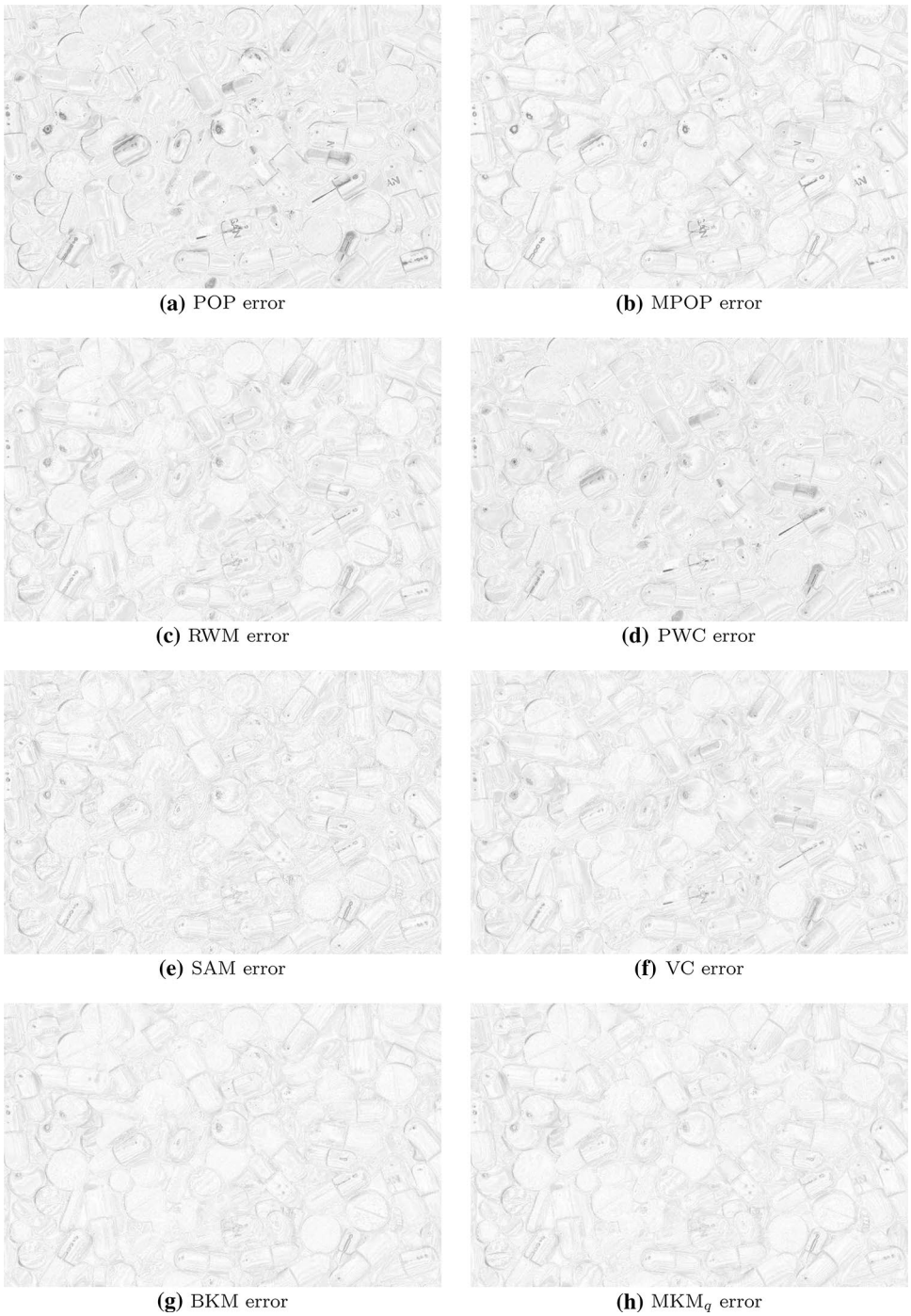
Fig. 7 Baboon error images ($K = 32$)

better visualization, pixel values of the error images were multiplied by 4 and then negated. It can be seen that the proposed MKM method performs exceptionally well in allocating representative colors to various image regions, resulting in minimal distortion (false contours, flat regions, color shifts, and color loss) and consequently cleaner error images. Together with the MSE results given in Table 2, these output images indicate that the proposed method and the well-known BKM method produce very similar results both quantitatively (numerically) and qualitatively (visually). The proposed method is significantly faster than the BKM method as the former is a one-pass approach, whereas the latter can require a few hundred passes to converge. Both methods are, however, easier to implement compared to many preclustering/postclustering methods.

4 Conclusions and future work

In this paper, an effective and efficient CQ method[3] was introduced. The method comprises cluster center initialization using the maximin method followed by color clustering using MacQueen's k-means algorithm. Detailed experiments on a large set of test images showed that the proposed method outperforms a large number of well-known CQ methods with respect to error minimization. The presented method is easy to implement and very efficient (requiring a fraction of a second to quantize a 512×512 image to 256 colors), exhibiting a sublinear scaling behavior in the number of colors. The method involves two adjustable parameters: exponent of the learning rate function and the sampling fraction, which respectively control the rate of learning and the proportion of input pixels used for learning. Unlike the parameters of many existing competitive learning-based CQ methods, these two parameters are not only intuitive, but also easy to set (setting them to their theoretically optimal values almost always gives the best results in practice). Future work includes further acceleration of the proposed CQ method by substituting the maximin initialization method with a more efficient alternative without sacrificing effectiveness or forgoing determinism.

[3] The source code of the proposed method is available at https://github.com/skythomp16/MacQueen-Color-Quantization



(a) POP error **(b)** MPOP error

(c) RWM error **(d)** PWC error

(e) SAM error **(f)** VC error

(g) BKM error **(h)** MKM_q error

Fig. 8 Pills error images ($K = 128$)

References

1. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with Bregman divergences. J. Mach. Learn. Res. **6**, 1705–1749 (2005)
2. Bermejo, S., Cabestany, J.: The effect of finite sample size on online K-means. Neurocomputing **48**(1–4), 511–539 (2002)
3. Bottou, L.: Online learning and stochastic approximations. In: Saad, D. (ed.) On-Line Learning in Neural Networks, pp. 9–42. Cambridge University Press, Cambridge (1998)
4. Bratley, P., Fox, B.L.: Algorithm 659: implementing Sobol's quasirandom sequence generator. ACM Trans. Math. Softw. **14**(1), 88–100 (1988)
5. Braudaway, G.W.: 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 (1987)
6. Brun, L., Mokhtari, M.: Two high speed color quantization algorithms. In: Proceedings of the 1st International Conference on Color in Graphics and Image Processing, pp. 116–121 (2000)

7. Brun, L., Trémeau, A.: Color quantization. In: Sharma, G. (ed.) Digital Color Imaging Handbook, pp. 589–638. CRC Press, Boca Raton (2002)
8. Celebi, M.E., Wen, Q., Hwang, S.: An effective real-time color quantization method based on divisive hierarchical clustering. J. Real-Time Image Proc. **10**(2), 329–344 (2015)
9. Celebi, M.E.: Fast color quantization using weighted sort-means clustering. J. Opt. Soc. Am. A **26**(11), 2434–2443 (2009)
10. Celebi, M.E.: Improving the performance of K-means for color quantization. Image Vis. Comput. **29**(4), 260–271 (2011)
11. Celebi, M.E., Wen, Q., Schaefer, G., Zhou, H.: Batch neural gas with deterministic initialization for color quantization. In: Proceedings of the International Conference on Computer Vision and Graphics, pp. 48–54 (2012)
12. Celebi, M.E., Hwang, S., Wen, Q.: Color quantization using the adaptive distributing units algorithm. Imaging Sci. J. **62**(2), 80–91 (2014)
13. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the K-means clustering algorithm. Expert Syst. Appl. **40**(1), 200–210 (2013)
14. Celebi, M.E. (ed.): Partitional Clustering Algorithms. Springer, Berlin (2015)
15. Cheng, S., Yang, C.: Fast and novel technique for color quantization using reduction of color space dimensionality. Pattern Recogn. Lett. **22**(8), 845–856 (2001)
16. Chung, K.L., Huang, Y.H., Wang, J.P., Cheng, M.S.: Speedup of color palette indexing in self-organization of Kohonen feature map. Expert Syst. Appl. **39**(3), 2427–2432 (2012)
17. Dekker, A.: Kohonen neural networks for optimal colour quantization. Network Comput. Neural Syst. **5**(3), 351–367 (1994)
18. Darken, C., Moody, J.: Fast adaptive K-means clustering: some empirical results. In: Proceedings of the 1990 International Joint Conference on Neural Networks, vol. 2, pp. 233–238 (1990)
19. Du, Q., Wong, T.W.: Numerical studies of MacQueen's K-means algorithm for computing the centroidal Voronoi tessellations. Comput. Math. Appl. **44**(3–4), 511–523 (2002)
20. El-Said, S.A.: Image quantization using improved artificial fish swarm algorithm. Soft. Comput. **19**(9), 2667–2679 (2015)
21. Frackiewicz, M., Palus, H.: KM and KHM clustering techniques for colour image quantisation. In: Tavares, J.M.R.S., Jorge, R.M.N. (eds.) Computational Vision and Medical Image Processing: Recent Trends. Springer, pp. 161–174 (2011)
22. Gervautz, M., Purgathofer, W.: A simple method for color quantization: octree quantization. In: Magnenat-Thalmann, N., Thalmann, D. (eds.) New Trends in Computer Graphics. Springer, pp. 219–231 (1988)
23. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theor. Comput. Sci. **38**(2–3), 293–306 (1985)
24. Gonzalez, A.I., Grana, M., Albizuri, F.X., D'Anjou, A., Torrealdea, F.J.: A near real-time evolution-based adaptation strategy for dynamic color quantization of image sequences. Inf. Sci. **122**(2–4), 161–183 (2000)
25. Grossberg, S.: Competitive learning: from interactive activation to adaptive resonance. Cogn. Sci. **11**(1), 23–63 (1987)
26. Hathaway, R.J., Bezdek, J.C., Huband, J.M.: Maximin initialization for cluster analysis. In: Proceedings of the 11th Iberoamerican Congress in Pattern Recognition, pp. 14–26 (2006)
27. Heckbert, P.: Color image quantization for frame buffer display. ACM SIGGRAPH Comput. Graph. **16**(3), 297–307 (1982)
28. Hsu, W.L., Nemhauser, G.L.: Easy and hard bottleneck location problems. Discret. Appl. Math. **1**(3), 209–215 (1979)
29. Hu, Y.C., Lee, M.G.: K-means based color palette design scheme with the use of stable flags. J. Electron. Imaging **16**(3), 033003 (2007)
30. Hu, Y.C., Su, B.H.: Accelerated K-means clustering algorithm for colour image quantization. Imaging Sci. J. **56**(1), 29–40 (2008)
31. Hu, Z., Su, Q., Xia, X.: Multiobjective image color quantization algorithm based on self-adaptive hybrid differential evolution. Comput. Intell. Neurosci. **2016**, 2450431 (2016)
32. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
33. Joy, G., Xiang, Z.: Center-cut for color image quantization. Vis. Comput. **10**(1), 62–66 (1993)
34. Lemire, D.: Fast random integer generation in an interval. ACM Trans. Model. Comput. Simul. **29**(1), 3:1–3:12 (2019)
35. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–136 (1982)
36. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
37. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)
38. Müllner, D.: Fastcluster: fast hierarchical, agglomerative clustering routines for R and Python. J. Stat. Softw. **53**(9), 1–18 (2013)
39. Omran, M.G.H., Engelbrecht, A.P., Salman, A.: Particle swarm optimization for pattern recognition and image processing. In: Abraham, A., Grosan, C., Ramos, V. (eds.) Swarm Intelligence in Data Mining, pp. 125–151. Springer, Berlin (2006)
40. Orchard, M., Bouman, C.: Color quantization of images. IEEE Trans. Signal Process. **39**(12), 2677–2690 (1991)
41. Ozturk, C., Hancer, E., Karaboga, D.: Color image quantization: a short review and an application with artificial bee colony algorithm. Informatica **25**(3), 485–503 (2014)
42. Nolle, L., Schaefer, G.: Colour map design through optimization. Eng. Optim. **39**(3), 327–343 (2007)
43. Perez-Delgado, M.L.: Colour quantization with ant-tree. Appl. Soft Comput. **36**, 656–669 (2015)
44. Perez-Delgado, M.L., Gallego, J.A.R.: A two-stage method to improve the quality of quantized images. J. Real-Time Image Process. (2019) **(in press)**
45. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes, 3rd edn. Cambridge University Press, Cambridge (2007)
46. Rumelhart, D.E., Zipser, D.: Feature discovery by competitive learning. Cogn. Sci. **9**(1), 75–112 (1985)
47. Schaefer, G., Zhou, H.: Fuzzy clustering for colour reduction in images. Telecommun. Syst. **40**(1–2), 17–25 (2009)
48. Schaefer, G., Nolle, L.: Color quantization. In: Qing, A. (ed.) Differential Evolution: Fundamentals and Applications in Electrical Engineering, pp. 399–405. Wiley, New York (2009)
49. Schaefer, G., Nolle, L.: A hybrid color quantization algorithm incorporating a human visual perception model. Comput. Intell. **31**(4), 684–698 (2015)
50. Schaefer, G., Hu, Q., Huiyu, Z., Peters, J.F., Hassanien, A.E.: Rough C-means and fuzzy rough C-means for colour quantisation. Fundam. Inf. Emerg. Comput. **119**(1), 113–120 (2012)
51. Scheunders, P.: A genetic C-means clustering algorithm applied to color image quantization. Pattern Recogn. **30**(6), 859–866 (1997)
52. Scitovski, R., Sabo, K.: Analysis of the K-means algorithm in the case of data points occurring on the border of two or more clusters. Knowl.-Based Syst. **57**, 1–7 (2014)
53. Spall, J.C.: Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. Wiley, Oxford (2003)
54. Su, Q., Hu, Z.: Color image quantization algorithm based on self-adaptive differential evolution. Comput. Intell. Neurosci. **2013**, 231916 (2013)

55. Uchiyama, T., Arbib, M.: An algorithm for competitive learning in clustering problems. Pattern Recogn. **27**(10), 1415–1421 (1994)
56. Ueda, Y., Koga, T., Suetake, N., Uchino, E.: Color quantization method based on principal component analysis and linear discriminant analysis for palette-based image generation. Opt. Rev. **24**(6), 741–756 (2017)
57. Valenzuela, G., Celebi, M.E., Schaefer, G.: Color quantization using coreset sampling. In: Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2096–2101 (2018)
58. Velho, L., Gomez, J., Sobreiro, M.V.R.: Color image quantization by pairwise clustering. In: Proceedings of the 10th Brazilian Symposium on Computer Graphics and Image Processing, pp. 203–210 (1997)
59. Wan, S.J., Prusinkiewicz, P., Wong, S.K.M.: Variance-based color image quantization for frame buffer display. Color Res. Appl. **15**, 52–58 (1990)
60. Wen, Q., Celebi, M.E.: Hard versus fuzzy C-means clustering for color quantization. EURASIP J. Adv. Signal Process. **2011**, 118–129 (2011)
61. Wu, X.: Efficient statistical computations for optimal color quantization. In: Arvo, J. (ed.) Graphics Gems, vol. II, pp. 126–133. Academic Press, London (1991)
62. Wu, K.L., Yang, M.S.: Alternative learning vector quantization. Pattern Recogn. **39**(3), 351–362 (2006)
63. Xiang, Z.: Color image quantization by minimizing the maximum intercluster distance. ACM Trans. Graph. **16**(3), 260–276 (1997)
64. Xiao, Y., Leung, C.S., Lam, P.M., Ho, T.Y.: Self-organizing map-based color palette for high-dynamic range texture compression. Neural Comput. Appl. **21**(4), 639–647 (2012)
65. Yair, E., Zeger, K., Gersho, A.: Competitive learning and soft competition for vector quantizer design. IEEE Trans. Signal Process. **40**(2), 294–309 (1992)
66. Yang, C.Y., Lin, J.C.: RWM-cut for color image quantization. Comput. Graph. **20**(4), 577–588 (1996)

M. Emre Celebi received his B.Sc. degree in Computer Engineering from the Middle East Technical University (Ankara, Turkey) in 2002. He received his M.Sc. and Ph.D. degrees in Computer Science and Engineering from the University of Texas at Arlington (Arlington, TX, USA) in 2003 and 2006, respectively. He is currently a Professor and the Chair of the Department of Computer Science at the University of Central Arkansas.

Dr. Celebi has actively pursued research in image processing/analysis and data mining with an emphasis on medical image analysis, color image processing, and partitional clustering. He has worked on several projects funded by the US National Science Foundation and National Institutes of Health and published nearly 150 articles in reputable journals and conference proceedings.

Dr. Celebi is an editorial board member of four international journals, reviews for over 110 international journals, and served on the program committee of more than 130 international conferences. He has been invited as speaker to several colloquia, workshops, and conferences, is the organizer of several workshops, and the editor of several journal special issues, books, and book series. He is a senior member of the IEEE and SPIE.