IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 18, 2019, accepted August 23, 2019, date of publication August 27, 2019,
date of current version September 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937934

A Hybrid Color Quantization Algorithm That Combines the Greedy Orthogonal Bi-Partitioning Method With Artificial Ants

MARÍA-LUISA PÉREZ-DELGADO AND JESÚS ÁNGEL ROMÁN GALLEGO
Escuela Politécnica Superior de Zamora, University of Salamanca, 49022 Zamora, Spain

Corresponding author: María-Luisa Pérez-Delgado (mlperez@usal.es)

This work was supported by the Samuel Solórzano Barruso Memorial Foundation of the University of Salamanca.

ABSTRACT A color quantization technique that combines the operations of two existing methods is proposed. The first method considered is the Greedy orthogonal bi-partitioning method. This is a very popular technique in the color quantization field that can obtain a solution quickly. The second method, called Ant-tree for color quantization, was recently proposed and can obtain better images than some other color quantization techniques. The solution described in this article combines both methods to obtain images with good quality at a low computational cost. The resulting images are always better than those generated by each method applied separately. In addition, the results also improve those obtained by other well-known color quantization methods, such as Octree, Median-cut, Neuquant, Binary splitting or Variance-based methods. The features of the proposed method make it suitable for real-time image processing applications, which are related to many practical problems in diverse disciplines, such as medicine and engineering.

INDEX TERMS Artificial intelligence, clustering methods, image processing.

I. INTRODUCTION

Nowadays images are very important elements in everyday communication. Social networks, web pages, reports, e-books or electronic documents include many images that must be stored, transmitted and displayed. Current devices can display high quality images with many colors. Nevertheless, the quality of the image is a disadvantage for its storage and transmission, since more colors mean more quality and also involve more storage space and less speed of transmission.

Color quantization reduces the number of different colors of an image trying to make the resulting image as similar as possible to the original. Reducing the colors of an image not only allows it to be displayed on low-end devices, but also reduces the size of the image and this allows it to be stored and transmitted more efficiently [1], [2]. Moreover, color quantization is also related to other processes applied to images, such as content-based image retrieval [3]–[5], texture analysis [6]–[8], image segmentation [9]–[13] and image watermarking [14], [15].

The associate editor coordinating the review of this article and approving it for publication was Dezhong Peng.

The color quantization problem is complex since the selection of the best colors to represent the image is an NP-complete problem [16]. Therefore, different solution approaches have been proposed for this problem. There are solution methods that work on the color cube and divide it into boxes. Some popular methods that apply this approach are the Greedy orthogonal bi-partitioning method (GOBP) [17], Octree [18], Median-cut [19], Binary splitting [20] and the Variance-based method [21]. Other solutions focus on the pixels of the image and separate them into groups. This is the approach applied by color quantization methods such as Neuquant [22] and Ant-tree for color quantization (ATCQ) [23], and also by other solutions that use non-specific methods of color quantization, such as K-means [24], Particle swarm optimization [25] and Artificial bee colony [26] algorithms.

In general, the methods of the first group are faster, but those of the second group can generate better images. Therefore, when selecting a color quantization method it will normally be necessary to choose between speed and quality, although it would be desirable to have both at the same time.

The GOBP method is a popular color quantization algorithm that applies an iterative process to divide the color space

into boxes, each of the resulting boxes defining a color of the quantized palette [17]. This method is very fast because the values used during the iterative process are calculated only once before the process begins. Certainly, this method is faster than most of the methods mentioned above.

The ATCQ method mimics the behavior of a set of ants that build a tree that allows an image to be represented using a reduced number of colors [23]. The ants, which represent the pixels of the image, connect to the tree taking into account the similarity among the colors they represent. Each subtree defines a color of the quantized palette and that color is used to represent all the ants of the subtree in the quantized image. This method was compared to other well-known color quantization methods, obtaining better images than Octree, Median-cut and Variance-based methods [23]. When compared to GOBP, ATCQ only generates better images in some cases, as reported in the article which proposed the second method. Nevertheless, it should be taken into account that GOBP generates better images than most of the color quantization methods mentioned above, as shown by the computational results included in several articles [27]–[33].

The objective of this article is to describe a method that combines the operations of GOBP and ATCQ in order to improve the quality of the resulting image, but without consuming too much time. Since both methods are applied sequentially, the hybrid method will only consume a little more time than ATCQ. Computational experiments show that the new method always generates better images than GOBP and ATCQ applied separately, although it consumes a little more time than ATCQ. GOBP is a deterministic method that always generates the same quantized image for each palette size. On the other hand, ATCQ can generate different quantized images depending on the values given to the parameters of the algorithm. Therefore, the result of GOBP will be used to define an initial solution to apply ATCQ, and the operations of this last method will be applied to improve said solution. GOBP has been chosen as starting point for two main reasons: it generates good quantized images (better than ATCQ in most cases) and is very fast. Both features allow to define a new rapid method that considers a good quantized palette as a starting point.

A drawback of splitting methods is that the splitting decision at each level can not be resumed. On the other hand, clustering-based methods are usually influenced by the initial conditions defined for the algorithms. The new method attempts to overcome both problems. To solve the first problem, it allows to assign each pixel p_i a color different from the centroid of the box associated with it by GOBP. To solve the second problem, it defines a good initial palette to apply the clustering-based method.

The rest of the article is organized as follows. First, the color quantization problem is defined (Section II) and the main solution methods are described (Section III). Next, Sections IV and V describe the two methods used to define the solution proposed in this article, which is presented in Section VI. After this, computational experiments are included in Section VII and the results of the tests are discussed. Finally, the conclusions of the article are presented.

II. THE COLOR QUANTIZATION PROBLEM

Let us consider a color image represented using the RGB color space. In this space, each pixel is defined by three integer values between 0 and 255 that represent the amount of red (R), green (G) and blue (B) associated with the pixel. Therefore, this color space allows to represent $256^3 = 16777216$ million different colors included in the RGB color cube (Fig. 1).

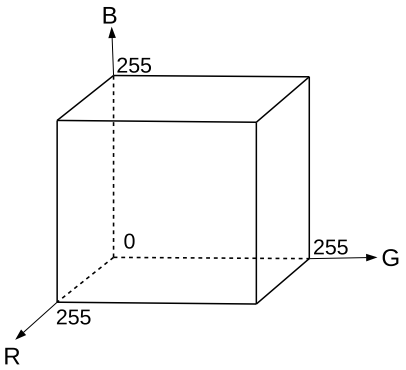


FIGURE 1. RGB color cube.

Let us consider an image with n pixels, $\{p_1, \ldots, p_n\}$, where $p_i = (R_i, G_i, B_i)$ represents the RGB color of a pixel, with $1 \leq i \leq n$. The color quantization process selects q colors to represent the pixels of the original image in such a way that the new image represented by those colors is as similar as possible to the original image. This process includes two operations. First, a quantized palette which includes the set of selected colors must be defined: $P = \{P_1, P_2, \ldots, P_q\}$, where each element P_j is an RGB color, $P_j = (R_j, G_j, B_j)$, and q is smaller than the number of colors of the original image. Next, this palette is used to associate a new color with each pixel of the original image, in order to obtain the quantized image. This last operation determines the color of each pixel p'_i of the quantized image, which corresponds to the pixel in the same position as p_i in the original image.

Therefore, the color quantization problem considers an original image whose pixels can take values from a palette with 256^3 different colors and defines a quantized palette with a limited number of colors, q, to represent the original image with minimal distortion.

III. OVERVIEW OF COLOR QUANTIZATION METHODS

The color quantization methods can be classified into two groups: splitting methods and clustering-based methods. Splitting methods apply an iterative process to divide the color space into boxes, until q boxes are obtained. Then, the quantized palette is defined including a representative color of each box. On the other hand, clustering-based methods separate the pixels of the image into groups or clusters and all the pixels in the same cluster are represented in

the quantized image by the same color. In general, splitting methods are faster, but clustering-based methods can obtain better images.

Splitting methods differ basically in the box that is selected at each iteration to split it, the splitting point and the splitting axis. Some popular splitting methods are Median-cut, Variance-based method, Binary splitting, Octree and GOBP. The last method is described in detail in the next section.

The Median-cut method selects the box that contains more pixels and divides it along the longest axis at the median point. Each of the resulting boxes is represented in the quantized palette by its average color [19].

The Variance-based method selects the box with the largest weighted variance and divides it along the axis with the least weighted sum of projected variances at the point that minimizes the marginal squared error [21].

The Binary splitting method considers the box with the largest dominant eigenvalue and splits it along the principal axis of the selected box. In this case the splitting point is the projection of the centroid to the selected axis [20].

The approach used by the Octree method is different from the previous methods, since it uses a tree structure to perform the color quantization. This structure can include 8 children per node and 8 levels as a maximum, which allows to have $8^8 = 16777216$ million leaves. To build the tree, the pixels of the image are processed and a leaf is included to store each of the colors used in the image. To reduce the number of colors to q, the algorithm selects the leaves that represent colors that are very close in the color space; such leaves are replaced by a single node with the average color of the leaves [18].

Other solutions have been proposed in the literature based on the previous methods [29], [30], [34], [35].

In general, clustering-based methods take techniques defined to solve problems other than color quantization and apply them to solve this specific problem. Some methods of this type are K-means, Fuzzy c-means, neural networks and swarm-based algorithms.

K-means is the most popular clustering method and it has been applied to solve the color quantization problem in several articles [27], [36]–[40]. This method considers q initial centroids and applies an iterative process to try to improve them. When the color quantization problem is considered, each centroid represents a color of the quantized palette. Each iteration associates each pixel with the closest centroid and then takes the average color of each group of pixels as the new value of the centroid. The process can stop after a predefined number of iterations or when a predefined error is reached. It should be noted that this method is slow and the result is influenced by the initial centroids.

The Fuzzy c-means method is based on the same idea as K-means, but in this case the items are not associated with a single group. On the contrary, each item is associated with several clusters and a variable determines its degree of membership to each cluster. This method attempts to solve the problem that arises when the data includes overlapping clusters. Fuzzy c-means has been applied to color quantization in [28], [41], [42].

Neuquant is a color quantization method that uses a self-organizing neural network. This network includes q neurons that are trained with the pixels of the original image, so that the final weights of these neurons define the colors of the quantized palette [22]. Other authors consider similar types of neural networks to solve the color quantization problem [43]–[45].

Several swarm-based methods have been applied to color quantization. These methods imitate the behavior of a group of individuals who perform very simple operations separately, but as a group they can solve complex problems. Omran *et al.* applied the Particle swarm optimization method to the color quantization problem [46]. In this case, the swarm is formed by a set of particles that represent quantized palettes. These particles move in the search space to improve the quality of the palettes they represent. This solution also uses the K-means algorithm, which is applied to each particle in a probabilistic way. On the other hand, Ozturk *et al.* applied the Artificial bee colony algorithm to the color quantization problem [47]. This solution simulates a swarm of bees trying to solve the problem. There is a set of food sources that represent quantized palettes and the operations of the bees select the best palette. This method also applies K-means in a probabilistic way to the food sources. The ATCQ method, based on the behavior of natural ants, also belongs to this family of methods and is described in Section V.

Recently, three color quantization methods based on ATCQ have been proposed: ITATCQ [31], ATCQ + FA [32] and BS + ATCQ [33]. ITATCQ applies the ATCQ operations iteratively in order to improve the quality of the quantized palette. The first iteration builds a 3-level tree by applying the same operations as ATCQ. To apply the next iteration, two operations are performed: first, all the ants are moved back to the root node; next, all the ants are reconnected to the tree by applying the ATCQ operations, but these operations are applied to a tree that includes not only the root node but also nodes in the second level. ATCQ + FA combines the Firefly algorithm [48] with the ATCQ algorithm. In this case, the position of each firefly of the swarm represents a quantized palette. In addition, there is a tree of ants associated with each firefly. Such tree is defined by the ATCQ algorithm and is used to refine the position of a firefly and to compute its quality. The initial position of each firefly is defined by applying ATCQ. Next, an iterative process combines the operations of ATCQ algorithm and Firefly algorithm. At each operation, the fireflies are first moved in the search space and then ATCQ is applied to refine the solution associated with each firefly and to compute the quality of the new solution. BS + ATCQ combines the Binary splitting method with the ATCQ method. Since both methods use a tree structure, the leaves of the binary tree defined by the first one are used to define the nodes in the second level of the tree used by the second one. After this, the ATCQ operations are applied to connect all the ants to the structure.

Algorithm 1 GOBP

1: Splitting process
2: Palette generation
3: Quantized image generation

Algorithm 2 Splitting Process – GOBP Algorithm

1: Build the histogram
2: Compute the moments
3: Set $k = 1$
4: **repeat**
5: Select the box with the largest variance, $T(c_l, c_m]$
6: Split this box into 2 boxes, $T(c_l, c'_m]$ and $T(c'_l, c_m]$, by a plane that minimizes the sum of variances of both sides
7: Set $k = k + 1$
8: **until** q boxes defined ($k = q$)

For a more detailed description of color quantization methods [49] can be consulted.

IV. THE GREEDY ORTHOGONAL BI-PARTITIONING METHOD

GOBP is a splitting method that selects the box with the largest weighted variance and divides it into two boxes along the axis that minimizes the sum of the variances on both sides of the cutting plane [17]. Algorithm 1 shows the main operations of this method. The first operation applies a splitting process to divide the color space into q boxes. The second operation defines the quantized palette including the average color of each box. The last operation of the algorithm uses this palette to define the quantized image.

The first step of GOBP algorithm includes several operations that are described in Algorithm 2. The first operation defines the histogram of the colors in the image, that is, the number of pixels of the image whose color is the same. Before defining the histogram, a pre-quantization is applied to reduce the number of bits used to represent each color. Instead of considering 8 bits per color, which defines a color space of $2^8 \times 2^8 \times 2^8 = 256 \times 256 \times 256$, the 3 least significant bits of each color component are discarded, so as the color space is reduced to a cube of $2^5 \times 2^5 \times 2^5 = 32 \times 32 \times 32$. This operation greatly reduces the number of different colors (from 256^3 to 32^3) and, consequently, the number of histogram entries. Therefore, for each pixel p_i of the original image, which is represented with 8 bits per color, the corresponding c_t value which includes only 5 bits per color is defined, where the values R_t, G_t and B_t that define c_t can take integer values between 0 and 31 (Fig. 2a). After this, the frequency of each c_t, $P(c_t)$, is determined.

To perform the splitting process in an efficient way, the second operation of Algorithm 2 calculates and stores several values that will be used in subsequent operations. For each point c_t of the $32 \times 32 \times 32$ cube defined in the previous step, three moments are computed by (1), (2), and (3), where $T(c_l, c_m]$ is the rectangular box defined by the points $c_l = (R_l, G_l, B_l)$ and $c_m = (R_m, G_m, B_m)$, with $R_l < r \leq R_m$, $G_l < g \leq G_m$ and $B_l < b \leq B_m$ (Fig. 2b), and o is a reference point such that $\sum_{c \in T(-\infty, o]} P(c) = 0$.

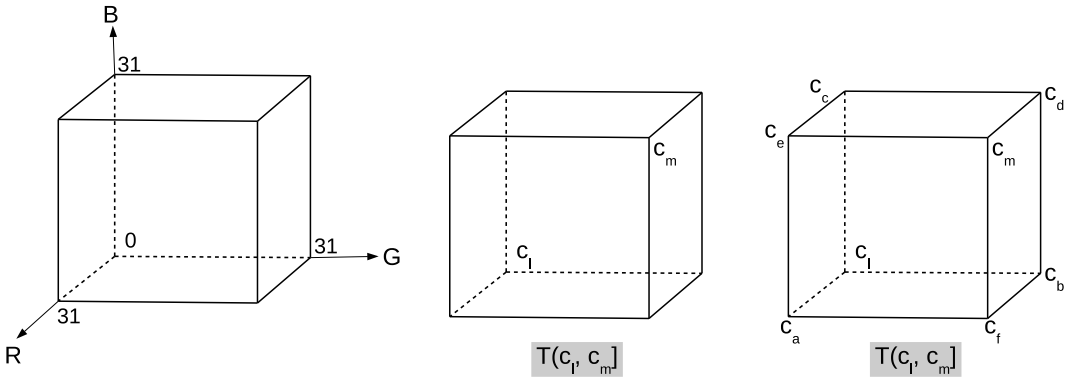
$$M_0(c_t) = \sum_{c \in T(o, c_t]} P(c) \tag{1}$$

$$M_1(c_t) = \sum_{c \in T(o, c_t]} cP(c) \tag{2}$$

$$M_2(c_t) = \sum_{c \in T(o, c_t]} cc^T P(c) \tag{3}$$

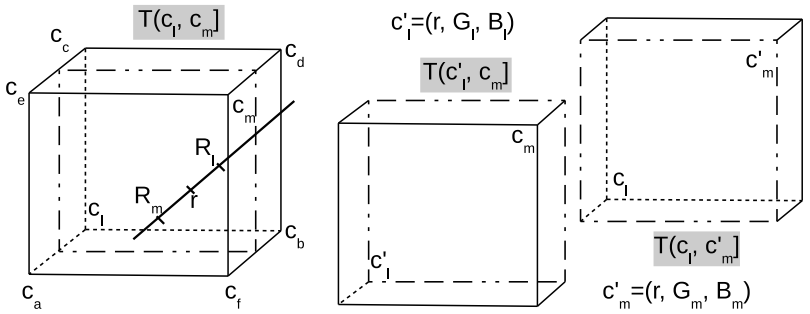
Once the moments have been computed, an iterative process is applied that selects a box and splits it into two boxes, concluding the iterations when q boxes have been defined. The box selected at each iteration is the one with the maximum variance among all the boxes defined until such moment. It should be noted that the only box existing at the beginning of this process is the $32 \times 32 \times 32$ cube (Fig. 2a).

To perform the splitting of the selected box at a good position, an orthogonal cutting plane is moved along each of the three dimensions of said box. The best cutting plane to perform the division is the one that minimizes the sum of the weighted variances of both sides of the plane.



(a) Initial box (b) General box (c) General box with all the vertex

FIGURE 2. (a) Initial box for the iterative process of GOBP method. (b) Box associated with points c_l and c_m. (c) All the vertex of the box associated with points c_l and c_m.



(a) cutting plane perpendicular to the R axis

(b) cutting plane perpendicular to the G axis

(c) cutting plane perpendicular to the B axis

FIGURE 3. The box $T(c_l, c_m]$ can be split into two boxes, $T(c_l, c'_m]$ and $T(c'_l, c_m]$ by a cutting plane passing through the pixel c'_l and that is perpendicular to the R axis (a), the G axis (b) or the B axis (c).

If $T(c_l, c_m]$ represents the selected box, three values are computed for said box: the pixel population (4), the mean (5) and the weighted variance (6).

$$w(c_l, c_m] = \sum_{c \in T(c_l, c_m]} P(c) \tag{4}$$

$$\mu(c_l, c_m] = \frac{\sum_{c \in T(c_l, c_m]} cP(c)}{w(c_l, c_m]} \tag{5}$$

$$E(c_l, c_m] = \sum_{c \in T(c_l, c_m]} cc^T P(c) - \frac{\left(\sum_{c \in T(c_l, c_m]} cP(c)\right)^2}{w(c_l, c_m]} \tag{6}$$

The values of w, μ and E can be calculated quickly from the moments computed and stored in the previous stage of the algorithm. With this objective, the rule of inclusion-exclusion of combinatorics [50] is applied, obtaining (7), where $f(c)$ may be 1, c or cc^T and the 8 vertex of the box $T(c_l, c_m]$ are considered as shown in Fig. 2c.

$$\begin{aligned} \sum_{c \in T(c_l, c_m]} f(c)P(c) = (& \sum_{c \in T(o, c_m]} + \sum_{c \in T(o, c_a]} + \sum_{c \in T(o, c_b]} + \sum_{c \in T(o, c_c]} \\ & - \sum_{c \in T(o, c_d]} - \sum_{c \in T(o, c_e]} - \sum_{c \in T(o, c_f]} \\ & - \sum_{c \in T(o, c_l]}) f(c)P(c) \end{aligned} \tag{7}$$

To determine the position of the cutting plane used to split the selected box, it is required to minimize $E(c_l, c'_m] + E(c'_l, c_m]$, where c'_l and c'_m can take one of the following values:

- $c'_l = (r, G_l, B_l)$ and $c'_m = (r, G_m, B_m)$, with $R_l < r \leq R_m$, if the cutting plane is perpendicular to the R axis (Fig. 3a).

- $c'_l = (R_l, g, B_l)$ and $c'_m = (R_m, g, B_m)$, with $G_l < g \leq G_m$, if the cutting plane is perpendicular to the G axis (Fig. 3b).
- $c'_l = (R_l, G_l, b)$ and $c'_m = (R_m, G_m, b)$, with $B_l < b \leq B_m$, if the cutting plane is perpendicular to the B axis (Fig. 3c).

The expression to be minimized is given by (8).

$$E(c_l, c'_m] + E(c'_l, c_m] = \sum_{c \in T(c_l, c_m]} c^2 P(c) - \frac{\left(\sum_{c \in T(c_l, c'_m]} cP(c)\right)^2}{w(c_l, c'_m]} - \frac{\left(\sum_{c \in T(c'_l, c_m]} cP(c)\right)^2}{w(c'_l, c_m]} \quad (8)$$

Wu indicated that minimizing (8) is equivalent to maximizing (9), which can also be calculated faster than (8) using the previously stored moments. Therefore, (9) is computed for each possible cutting plane and the one that generates the maximum value is used to split the selected box. This results in two new boxes, $T(c_l, c'_m]$ and $T(c'_l, c_m]$, and the current iteration of the algorithm concludes.

$$\frac{\left(\sum_{c \in T(c_l, c'_m]} cP(c)\right)^2}{w(c_l, c'_m]} + \frac{\left(\sum_{c \in T(c'_l, c_m]} cP(c)\right)^2}{w(c'_l, c_m]} = \frac{\left(\sum_{c \in T(c_l, c'_m]} cP(c)\right)^2}{w(c_l, c'_m]} + \frac{\left(\sum_{c \in T(c_l, c_m]} cP(c) - \sum_{c \in T(c_l, c'_m]} cP(c)\right)^2}{w(c_l, c_m] - w(c_l, c'_m]} \quad (9)$$

Algorithm 3 Palette Generation – GOBP Algorithm

1: Set $k = 1$
2: **for** each box $T(c_l, c_m]$ of the partition **do**
3: Set $P_k = \mu(c_l, c_m]$
4: Set $tag(c_t) = k \; \forall \; c_t \in T(c_l, c_m]$
5: Set $k = k + 1$
6: **end for**

When the splitting process ends, the quantized palette is defined (Algorithm 3). To do this, each of the boxes obtained is processed and its average value is taken as a color of the palette. In addition, each point c_t of a box is associated with a label that identifies the color of the quantized palette defined by said box ($tag(c_t) = k$). The last operation of the algorithm uses these labels to define the quantized image.

To define the final image, the pixel p'_i of the quantized image corresponding to each pixel p_i of the original image must be defined (Algorithm 4). To set the value of p'_i, it is first necessary to determine the value c_t associated with the pixel p_i in the color cube of size $32 \times 32 \times 32$. Once this value has been determined, the color assigned to p'_i is the color of the quantized palette defined by the box containing c_t, which is identified by $tag(c_t)$.

Algorithm 4 Quantized Image Generation – GOBP Algorithm

1: **for** each pixel p_i of the original image **do**
2: Determine the corresponding point in the $32 \times 32 \times 32$ color cube, c_t
3: Set $k = tag(c_t)$
4: Set $p'_i = P_k$
5: **end for**

V. THE ANT-TREE FOR COLOR QUANTIZATION METHOD

The ATCQ algorithm is a color quantization method based on the Ant-tree algorithm [23]. It has been observed that some species of ants can avoid obstacles or cross empty spaces by connecting their bodies to build structures. The Ant-tree algorithm tries to mimic this self-assembly behavior to solve clustering problems [51], [52].

To solve the color quantization problem by ATCQ, each pixel p_i of the original image is represented by an ant h_i. The operations of the algorithm connect the ants in a tree structure taking into account the similarity among the pixels they represent. To define this tree, three types of nodes are used. The root node of the tree, a_0, is at the top level of the structure and is called support. The children of the support, $\{S_1, \ldots, S_q\}$, are in the second level of the structure. The other levels of the structure include the ants when they connect to the tree.

Each node S_j is the root of a subtree that defines a cluster of ants and also defines a color of the quantized palette. For this purpose, there are three values associated with S_j: the number of ants connected to the subtree, nc_j, the sum of the RGB colors of such ants, sum_j, and the sum of the similarities between each ant h_i connected to the subtree j and the color of such subtree when the ant was included in it, e_j. The color of the subtree is calculated based on the previous values: $color_j = sum_j / nc_j$. The three values associated with node S_j are initialized when this node is created and then they are updated when new ants are included in the subtree j.

The number of subtrees (clusters) defined by the algorithm is equal to the value of q, with $0 \leq q \leq Q_{max}$, where Q_{max} defines the maximum number of colors in the quantized palette. At the beginning of the algorithm, the tree only includes the root node and q is 0. As the operations of the algorithm progress, new subtrees are defined, so q increases. When q takes the value Q_{max}, no more subtrees can be created, since the palette already includes the maximum number of colors allowed. Therefore, Q_{max} limits the number of children of the support. Another limit, L_{max}, can be associated with the remaining nodes of the tree, which may be different from Q_{max}.

Algorithm 5 shows the main operations of ATCQ. At the beginning of the algorithm the tree only includes the root node, where all the ants are waiting to connect to the structure. Each iteration of the algorithm takes a moving ant h_i (that is, an ant not yet connected to the tree) which is on some node, denoted a_{pos}. The operations applied to h_i depend on its

Algorithm 5 ATCQ

1: **while** there are moving ants **do**
2: Take a moving ant, h_i, now on a_{pos}
3: **if** $(a_{pos} = a_0)$ **then**
4: Support case operations
5: **else**
6: Not-support case operations
7: **end if**
8: **end while**
9: **for** each subtree k with root in the second level **do**
10: Set $p'_i = color_k \ \forall \ h_i$ in the subtree
11: **end for**

Algorithm 6 Support Case Operations – ATCQ Algorithm

1: **if** $(q = 0)$ **then**
2: Create a new child of a_0 and connect h_i
3: **else**
4: Select the subtree c with the color most similar to h_i. Let $d_{ic} = \text{Sim}(h_i, color_c)$
5: **if** $(d_{ic} < T_c)$ **and** $(q < Q_{max})$ **then**
6: Create a new child of a_0 and connect h_i
7: **else**
8: Update the subtree with root in S_c
9: **end if**
10: **end if**

position on the tree: if it is on the support node ($a_{pos} = a_0$), the operations defined in Algorithm 6 are applied, otherwise those defined in Algorithm 7 are applied. The iterations of ATCQ end when all the ants have been connected to the structure and at this moment the final tree includes q subtrees. The color of each subtree defines an element of the quantized palette, $P = \{color_1, color_2, \ldots, color_q\}$. In addition, the color of each subtree is used to represent in the quantized image all the ants (pixels of the original image) included in the subtree.

Algorithm 6 considers two situations with different treatment:

- if the tree only includes the root node ($q = 0$), the first child of that node is created, S_1, and then the selected ant is connected as the first child of S_1. This situation only occurs when the first ant is processed.
- if the tree includes at least one subtree ($q > 0$), h_i can move to an existing subtree or to a new subtree. To make a decision, the node S_c in the second level of the tree with the color most similar to h_i is determined. Let d_{ic} denote the similarity between h_i and $color_c$, whose value is computed by using the Euclidean metric.
  If the similarity between h_i and the color of the subtree c reaches a threshold T_c ($d_{ic} \geq T_c$), such subtree is updated to include the ant h_i; if the similarity does not reach the threshold, a new child of a_0 is created to connect h_i (if a_0 cannot accept more children, h_i is included in the subtree c). The threshold T_c is calculated by (10), where α is a parameter in (0, 1].

$$T_c = \frac{e_c}{nc_c}\alpha \tag{10}$$

To create a new child of a_0, the current value of q is incremented by 1 and then the node S_q is created with the following initial values: $nc_q = 1$, $sum_q = h_i$, $e_q = 0$. The operation concludes when h_i connects as the first child of S_q.

To update the subtree with root in node S_c, the ant h_i is placed on S_c. In addition, the three values associated with this node are updated, to indicate that a new ant has been included in the subtree: $nc_c = nc_c + 1$, $sum_c = sum_c + h_i$, $e_c = e_c + d_{ic}$.

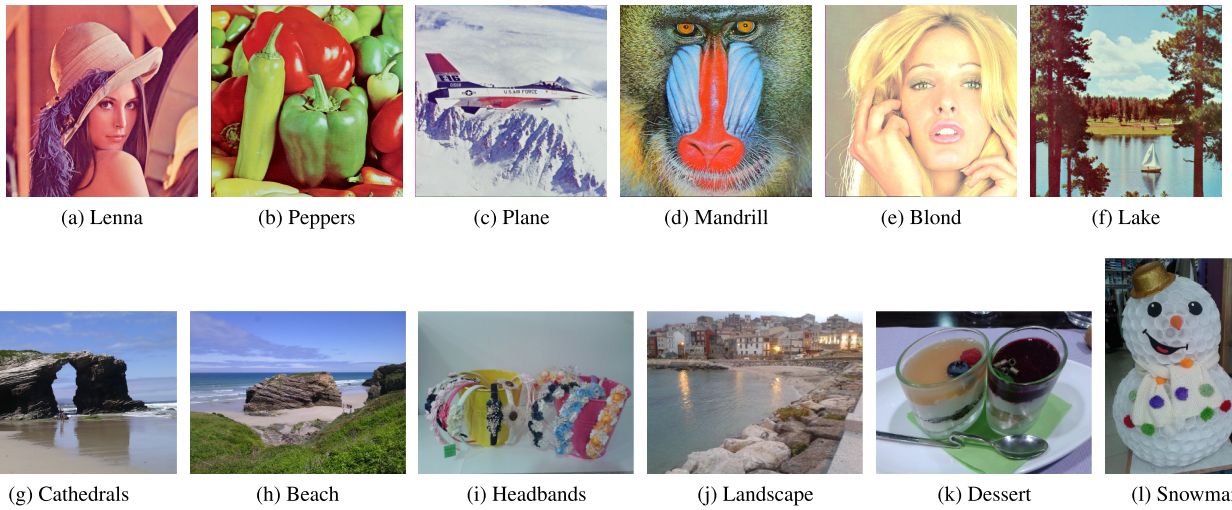
Algorithm 7 Not-Support Case Operations – ATCQ Algorithm

1: **if** no child connected to a_{pos} **then**
2: Connect h_i to a_{pos}
3: **else if** two ants connected to a_{pos} and the second one never disconnected **then**
4: Move to a_0 the subtree with root in the second child of a_{pos}
5: Connect h_i to a_{pos}
6: **else**
7: Select a node connected to a_{pos}, a^+
8: **if** $(\text{Sim}(h_i, a^+) < T_c)$ **and** (a_{pos} has less than L_{max} children) **then**
9: Connect h_i to a_{pos}
10: **else**
11: Move h_i towards a^+
12: **end if**
13: **end if**

Algorithm 7 considers three situations with different treatment:

- If a_{pos} has no child, the selected ant connects as its first child.
- If a_{pos} has two children and the second one has never been disconnected from a_{pos}, two operations are performed. First, the subtree with root in the second child is disconnected from the structure and all the ants in this subtree are moved back to the support. Next, h_i connects as the new second child of a_{pos}.
- In other cases, a random child of a_{pos}, denoted a^+, is selected. If the similarity between h_i and a^+ reaches the threshold T_c, the selected ant moves towards node a^+; in other case, such ant must connect as a new child of a_{pos}. Since there is a limit to the number of children a node can have, this connection is only possible if a_{pos} has not reached that limit; if a_{pos} already has L_{max} children, the connection is impossible and therefore h_i moves towards a^+.

The disconnection of ants performed in the second situation gives some ants a new opportunity to select a better position in the tree. However, the ATCQ algorithm can be simplified by eliminating the disconnection of ants (removing operations from line 3 to line 5 of Algorithm 7).



(a) Lenna (b) Peppers (c) Plane (d) Mandrill (e) Blond (f) Lake

(g) Cathedrals (h) Beach (i) Headbands (j) Landscape (k) Dessert (l) Snowman

FIGURE 4. Images used for the tests.

Algorithm 8 WATCQ

1: Splitting process
2: Initial tree definition
3: ATCQ

Computational results presented in [23] show that the resulting algorithm is faster, since each ant is processed only once, although the quantized images may be a bit worse than those obtained when disconnection is allowed.

VI. THE PROPOSED ALGORITHM

The method proposed in this article first applies GOBP operations and then applies ATCQ operations. The resulting algorithm, called WATCQ, discards some of the operations of both methods and reduces to three main steps (Algorithm 8):

- The first step defines a partition of the color cube by applying GOBP.
- The second step uses the result of the previous step to define an initial tree.
- The third step applies ATCQ operations to improve the result obtained in the first step. This operation also includes the generation of the quantized image.

WATCQ does not require applying the GOBP operations to generate the quantized palette or to define the quantized image. The new algorithm only needs the partition defined by GOBP, which is used to create initial nodes of the tree before applying the ATCQ operations.

The first operation of WATCQ is the same as that described for the GOBP algorithm in Section IV. As indicated in that section, this operation applies a splitting process that divides the color cube into q boxes and the average color of each box defines an element of the quantized palette. The boxes defined by GOBP are taken as the initial information to apply the ATCQ operations.

As described in Section V, ATCQ operates on a tree that initially only includes the root node and new nodes are

Algorithm 9 Initial Tree Definition – WATCQ Algorithm

1: Set $k = 1$
2: **for** each box $T(c_l, c_m]$ of the partition **do**
3: Create a new child of the support, S_k
4: Set $nc_k = w(c_l, c_m]$ and $sum_k = \mu(c_l, c_m]w(c_l, c_m]$
5: Compute e_k by (11)
6: Set $k = k + 1$
7: **end for**

Algorithm 10 Support Case Operations – WATCQ Algorithm

1: Select the subtree c with the color most similar to h_{ij}. Let $d_{ic} = \text{Sim}(h_i, color_c)$
2: Update the subtree with root in S_c

TABLE 1. Features of the test images: Name, width and height (pixels), number of different colors.

name	*width*	*height*	*colors*
Lenna	512	512	148279
Peppers	512	512	183525
Plane	512	512	77041
Mandrill	512	512	230427
Blond	512	512	79228
Lake	512	512	168459
Cathedrals	600	450	66305
Beach	600	450	124335
Headbands	600	450	93303
Landscape	600	450	93255
Dessert	600	450	103792
Snowman	450	600	92413

included as the operations of the algorithm progress. The second level of this tree can include Q_{max} nodes at most, and each time a node of this level is created a new element of the quantized palette is defined. However, when combining ATCQ with GOBP to define the WATCQ algorithm, the initial tree considered is different. For ATCQ operations to take advantage of the result obtained by GOBP, this result is

TABLE 2. ***MSE*** **results for GOBP, ATCQ without disconnection of ants (ATCQn), ATCQ with disconnection of ants (ATCQd), and both variants of the proposed method (WATCQn: Variant that considers ATCQ without disconnection of ants, WATCQd: Variant that considers ATCQ with disconnection of ants). (*q*: Number of colors of the quantized palette; MSE_m: Minimum *MSE*; MSE_a: Average *MSE*; *dev*: Standard deviation of *MSE*.)**

		GOBP	ATCQn			ATCQd			WATCQn	WATCQd		
	q	MSE	MSE_m	MSE_a	dev	MSE_m	MSE_a	dev	MSE	MSE_m	MSE_a	dev
Lenna	16	269.75	312.66	365.42	52.70	279.27	357.47	55.99	223.46	222.51	222.91	0.25
	32	158.61	163.55	209.00	45.94	155.48	213.45	40.98	131.41	131.07	131.27	0.11
	64	99.16	96.16	128.83	33.08	91.23	128.88	31.56	79.13	78.84	78.94	0.06
	128	61.79	58.87	93.05	47.16	58.94	93.81	43.11	49.54	49.58	49.64	0.03
	256	39.53	38.78	80.91	55.94	38.60	78.47	51.37	32.16	32.31	32.34	0.01
Peppers	16	479.62	556.68	1194.82	805.55	553.47	1725.56	1179.21	425.22	424.72	425.06	0.14
	32	279.28	334.17	421.08	79.90	328.49	748.24	515.48	241.08	240.53	240.70	0.12
	64	165.37	188.96	274.94	114.72	173.62	285.91	86.38	142.07	141.92	142.00	0.04
	128	102.31	114.89	224.60	152.47	110.20	213.53	120.88	88.64	88.65	88.72	0.03
	256	66.08	73.32	205.57	168.59	72.18	187.82	140.96	56.90	56.92	56.99	0.03
Plane	16	158.91	227.03	297.99	77.02	191.40	335.91	91.17	135.70	135.53	135.74	0.10
	32	85.45	116.71	174.35	56.03	104.89	184.81	70.69	65.63	65.58	65.67	0.06
	64	51.33	64.15	101.61	43.03	59.05	101.47	33.80	40.59	40.73	40.77	0.02
	128	32.60	35.49	78.45	51.40	34.77	68.63	24.65	25.20	25.28	25.31	0.01
	256	21.66	24.62	66.65	58.56	22.71	52.56	31.90	15.93	16.01	16.05	0.01
Mandrill	16	778.41	806.08	2242.10	3393.85	752.73	1327.46	1584.32	659.74	656.98	658.03	0.53
	32	468.39	538.49	2023.16	3501.72	491.87	927.06	1059.84	405.65	404.58	404.84	0.16
	64	288.33	362.35	1913.97	3557.12	326.37	1024.81	1748.67	249.00	248.57	248.76	0.10
	128	186.33	220.56	1837.06	3597.27	201.98	693.75	1110.63	162.45	162.04	162.21	0.07
	256	118.65	147.20	1797.99	3618.27	126.83	844.04	1741.77	102.82	102.75	102.83	0.03
Blond	16	202.06	224.43	284.55	66.15	217.35	291.53	54.69	172.02	171.71	171.98	0.07
	32	107.55	132.69	177.37	41.44	138.74	182.28	33.50	90.68	90.37	90.50	0.04
	64	63.03	97.80	120.36	20.77	89.90	120.74	21.64	51.46	51.42	51.49	0.04
	128	36.84	55.38	83.72	36.50	48.15	80.98	28.22	31.17	31.32	31.34	0.01
	256	23.59	28.07	69.28	46.94	26.66	65.62	38.61	19.89	20.02	20.05	0.01
Lake	16	381.68	438.22	843.69	315.68	388.27	917.41	260.10	328.01	327.42	327.67	0.10
	32	249.81	268.93	482.77	285.85	263.78	560.76	305.89	215.70	215.25	215.47	0.10
	64	161.34	182.78	222.66	48.81	180.68	228.16	40.02	139.07	138.80	138.97	0.05
	128	102.55	122.82	165.14	70.85	117.05	151.95	33.79	88.50	88.46	88.53	0.04
	256	66.47	78.16	142.53	87.38	73.35	123.27	54.71	57.26	57.31	57.35	0.02

used to define an initial tree and then ATCQ is applied. The second step of WATCQ, whose operations are detailed in Algorithm 9, defines this initial tree.

Algorithm 9 applies an iterative process to include q nodes in the second level of the tree, $S_1, S_2, \ldots, S_q$, and to set initial values for the parameters nc_k, sum_k and e_k associated with each node S_k. The values considered for these parameters are different from those described in Section V for the general ATCQ algorithm, because now the nodes of the second level are initially used to store information obtained by GOBP. A total of q iterations are performed to process all the boxes of the partition defined by GOBP. Each iteration processes a box $T(c_l, c_m]$ of the partition and defines a new child of the support node, S_k, where k takes values between 1 and q. The number of pixels associated with the subtree k is set to the number of pixels in the box ($nc_k = w(c_l, c_m]$). The color of the box $T(c_l, c_m]$ is taken as the color of the subtree k ($color_k = \mu(c_l, c_m]$) and sum_k is computed from the previous values: $sum_k = color_k nc_k$ (which is equivalent to $sum_k = \mu(c_l, c_m]w(c_l, c_m]$). The last parameter to be defined is e_k, which represents the sum of similarities between the color of each ant (pixel) associated with a subtree and the color of that subtree. To compute the initial value of this parameter, (11) is applied, where *Sim* represents the similarity computed using the Euclidean metric.

$$e_k = \sum_{c_j \in T(c_l, c_m]} Sim(c_j, \mu(c_l, c_m]) \tag{11}$$

At this point of the WATCQ algorithm, a tree has been defined with q nodes in the second level, each representing a color of the quantized palette defined by the GOBP algorithm. Once the initial tree has been built, ATCQ operations are applied to connect the ants to the tree. As in this case the initial tree already includes in the second level the maximum number of nodes allowed, the operations applied when the selected ant is on the support (Algorithm 6 – Support case operations) can be simplified. In this case new subtrees (new elements of the quantized palette) cannot be created and this forces to associate each ant with the most similar subtree. Therefore, Algorithm 6 is replaced with Algorithm 10 when WATCQ is applied.

It should be noted that the α parameter has a different effect on WATCQ and ATCQ. When the general ATCQ algorithm is considered, this parameter influences the decision to create a new node in the second level of the tree, that is, a new color of the quantized palette. Nevertheless, the initial tree considered by WATCQ already includes the maximum number of possible nodes in the second level. Therefore, in this

TABLE 3. *MSE* results for GOBP, ATCQ without disconnection of ants (ATCQn), ATCQ with disconnection of ants (ATCQd), and both variants of the proposed method (WATCQn: Variant that considers ATCQ without disconnection of ants, WATCQd: Variant that considers ATCQ with disconnection of ants). (q: Number of colors of the quantized palette; MSE_m: Minimum MSE; MSE_a: Average MSE; dev: Standard deviation of MSE.)

		GOBP	ATCQn			ATCQd			WATCQn	WATCQd		
	q	MSE	MSE_m	MSE_a	dev	MSE_m	MSE_a	dev	MSE	MSE_m	MSE_a	dev
Cathedrals	16	153.92	173.81	631.01	455.35	154.93	627.78	440.44	132.54	132.54	132.66	0.05
	32	81.90	93.29	318.32	327.10	80.46	300.96	268.99	63.25	63.17	63.24	0.04
	64	45.28	47.86	150.86	183.90	45.43	146.85	163.15	34.33	34.37	34.42	0.02
	128	27.31	32.40	58.29	30.69	27.42	56.41	26.95	20.31	20.48	20.52	0.02
	256	18.10	19.93	51.80	36.24	17.28	47.33	32.24	12.64	12.78	12.81	0.01
Beach	16	332.52	376.17	989.47	388.15	399.57	952.98	316.83	297.17	296.73	297.05	0.10
	32	177.34	268.19	632.84	387.64	243.41	598.49	321.15	145.64	145.20	145.32	0.09
	64	101.92	141.02	422.14	332.81	129.25	401.49	288.09	81.68	81.42	81.55	0.07
	128	59.64	82.81	186.92	121.53	71.83	221.86	208.35	47.07	47.02	47.07	0.02
	256	36.33	53.91	111.05	77.20	42.71	100.50	51.17	28.47	28.54	28.57	0.02
Headbands	16	246.65	297.69	614.50	327.53	261.90	648.37	283.01	212.43	211.67	211.89	0.14
	32	142.63	181.41	321.72	154.63	160.80	381.98	221.59	120.37	119.93	120.23	0.13
	64	87.52	100.66	184.64	83.56	94.61	200.06	93.07	70.63	70.55	70.62	0.03
	128	53.42	63.82	121.44	64.60	57.47	107.79	43.07	43.05	43.07	43.13	0.03
	256	33.83	34.52	90.06	70.18	34.23	73.02	44.05	26.17	26.25	26.29	0.02
Landscape	16	241.33	267.62	584.36	356.22	233.42	696.28	387.11	211.12	210.97	211.13	0.06
	32	131.31	124.61	287.08	216.55	113.94	315.68	192.55	105.55	105.16	105.25	0.05
	64	72.20	70.41	132.05	77.85	66.28	126.24	57.14	58.21	58.16	58.24	0.03
	128	42.75	42.25	93.70	79.52	41.96	64.55	19.12	33.37	33.44	33.48	0.02
	256	25.75	27.11	86.43	84.66	26.18	78.12	73.83	20.40	20.54	20.57	0.01
Dessert	16	299.46	292.09	395.75	74.89	293.07	442.49	170.83	252.94	252.32	252.59	0.13
	32	160.65	192.26	265.50	64.44	185.06	230.29	28.34	130.46	130.28	130.39	0.07
	64	90.42	123.10	175.44	83.77	105.35	155.43	37.72	71.30	71.09	71.16	0.04
	128	52.66	80.33	133.67	103.01	63.91	105.04	47.93	42.51	42.53	42.59	0.03
	256	32.71	41.75	117.02	113.95	36.28	87.45	62.53	25.36	25.43	25.46	0.01
Snowman	16	308.55	305.38	460.54	187.23	260.57	522.39	252.03	253.84	253.29	253.60	0.11
	32	161.12	161.19	200.38	24.02	164.58	221.03	41.19	124.82	124.58	124.81	0.11
	64	89.53	123.10	175.44	83.77	94.46	142.08	31.20	66.87	66.71	66.81	0.04
	128	49.69	57.94	97.52	59.64	54.07	97.65	43.59	37.75	37.66	37.74	0.04
	256	29.85	35.41	77.93	69.81	30.61	72.78	56.97	22.34	22.41	22.43	0.01

algorithm α only influences the operations performed at lower levels of the tree, where it is used to compute the threshold that determines whether an ant connects to the structure or not.

As the ants are processed and they connect to the structure, the parameters of the nodes in the second level of the tree are updated. When all the ants have been connected to the structure, the final colors of the nodes in the second level are defined not only by the colors of the ants (pixels) associated with each subtree, but also by the initial color assigned to the root of each subtree taken from the partition defined by GOBP.

The last operation of WATCQ algorithm defines the quantized image in the same way described for ATCQ in Section V.

Many color quantization methods combine previously existing techniques [27]–[30], [32]–[40], [46], [47]. These solutions take advantage of the characteristics of the methods that combine to obtain better images or to accelerate the operations. Although it is desirable to obtain good quality images quickly, this is difficult. Therefore, each method focuses on one of the two objectives. The solution proposed in this article tries to achieve the balance between quality and speed and this objective conditioned the methods used to define the new algorithm.

The ATCQ operations applied to the partition defined by GOBP improve the result of the hybrid method for two main reasons. First, these operations are applied to the pixels of the original image, so the distribution of these pixels in the color space is taken into consideration. Second, the value p'_i associated with each pixel p_i is the most similar in the palette, and this increases the similarity between the original and the quantized image. In addition, ATCQ updates the values of the initial quantized palette defined by GOBP, so that at the end of WATCQ operations the palette no longer represents the centroids of the boxes of the initial partition.

GOBP applies a pre-quantization to reduce the number of bits used to represent each color. This operation allows to reduce the size of the histogram defined in the first step of Algorithm 2 and also reduces the data set used to perform calculations. The operation is simple and fast, but it is independent of the image content. Since this pre-quantization does not take into account the distribution of colors of the original image, it only generates good results if the colors of the image are uniformly distributed in the RGB color cube [49], [53], [54]. Nevertheless, this distribution is not

TABLE 4. Execution time (milliseconds) for GOBP, ATCQ without disconnection of ants (ATCQn), ATCQ with disconnection of ants (ATCQd), and both variants of the proposed method (WATCQn: Variant that considers ATCQ without disconnection of ants, WATCQd: Variant that considers ATCQ with disconnection of ants). (q: Number of colors of the quantized palette.)

q		GOBP	ATCQn	ATCQd	WATCQn	WATCQd		GOBP	ATCQn	ATCQd	WATCQn	WATCQd
16	Lenna	14	39	189	60	231	Cathedrals	14	36	165	59	187
32		14	61	230	94	279		14	56	204	91	239
64		14	97	285	160	344		14	83	246	163	317
128		14	151	370	277	490		14	117	305	280	484
256		15	204	476	546	786		15	158	375	533	780
16	Peppers	14	37	179	55	214	Beach	14	36	161	61	215
32		14	60	222	89	261		14	54	190	93	254
64		14	97	278	152	332		14	84	231	158	324
128		14	150	366	276	473		14	126	307	280	478
256		15	215	481	524	729		15	170	382	532	778
16	Plane	14	37	207	58	219	Headbands	14	37	174	60	221
32		14	57	241	91	265		14	54	277	91	267
64		14	87	308	153	354		14	80	336	156	351
128		14	120	414	279	525		14	119	312	280	508
256		14	151	441	525	830		15	160	392	527	832
16	Mandrill	14	37	181	62	223	Landscape	14	39	178	61	208
32		14	56	220	94	260		14	60	218	93	265
64		14	83	278	147	342		14	98	279	173	350
128		15	126	327	255	494		14	139	377	283	498
256		15	179	459	453	770		15	192	518	532	819
16	Blond	14	39	186	60	226	Dessert	14	38	180	60	205
32		14	62	226	91	267		14	58	217	92	251
64		14	94	283	156	341		14	93	272	156	317
128		14	141	365	278	508		14	137	352	287	482
256		14	191	441	527	846		15	200	464	537	814
16	Lake	14	39	170	60	197	Snowman	14	39	175	61	215
32		14	64	209	93	239		14	63	214	92	264
64		14	104	306	157	320		14	94	283	158	342
128		14	167	398	284	469		14	157	378	283	496
256		15	228	517	514	799		15	218	493	535	794

common in real images. On the contrary, there are many images whose color distribution includes regions of the color space with many points and other regions not represented. When the pre-quantization is applied to these images, the interesting colors may not be sampled with sufficient density.

When WATCQ applies the ATCQ operations, they do not use the pre-quantized information but use all the pixels of the original image. In this way, the final part of the WATCQ algorithm takes into account the distribution of colors of the original image and this improves the result [49], [53], [54].

The color quantization methods apply two different strategies to associate a color of the quantized palette to each pixel of the image, in order to generate the quantized image. One strategy uses the centroid of each box or cluster to represent all the colors in that box or cluster. The other strategy associates each pixel of the original image with the closest color in the quantized palette. The first strategy is faster, but the second generates better images. GOBP applies the first strategy, so that each pixel p_i of the original image is mapped to the centroid of the box containing the value obtained after applying the pre-quantization operation to p_i. Therefore, p_i could be represented by a color of the quantized palette that is not the most appropriate.

ATCQ operations does not compute the colors of the palette as the centroids of the subtrees. It can be considered that they compute pseudo-centroids, since the values are computed as the iterations of the algorithm progress and take into account not only the color of all the ants connected to each subtree (even the color of the ants that were disconnected from the subtree) but also the centroid of a box of the partition defined by GOBP.

VII. RESULTS AND DISCUSSION

The effectiveness of the proposed model has been analyzed by applying it to two sets of color images, which are included in Fig. 4. The first set (images from (a) to (f)) includes images commonly used to analyze color quantization methods, available at [55], whereas the second set includes six images proposed by the authors of this article, which can be downloaded from [56]. Table 1 shows the features of all the images.

To determine the quality of the quantized image, the mean squared error (MSE) has been considered, since it is a measure commonly used in the color quantization literature. This error measure is computed by (12), where p_i represents the RGB color of a pixel of the original image and p'_i represents the color of the pixel in the same position, but in the

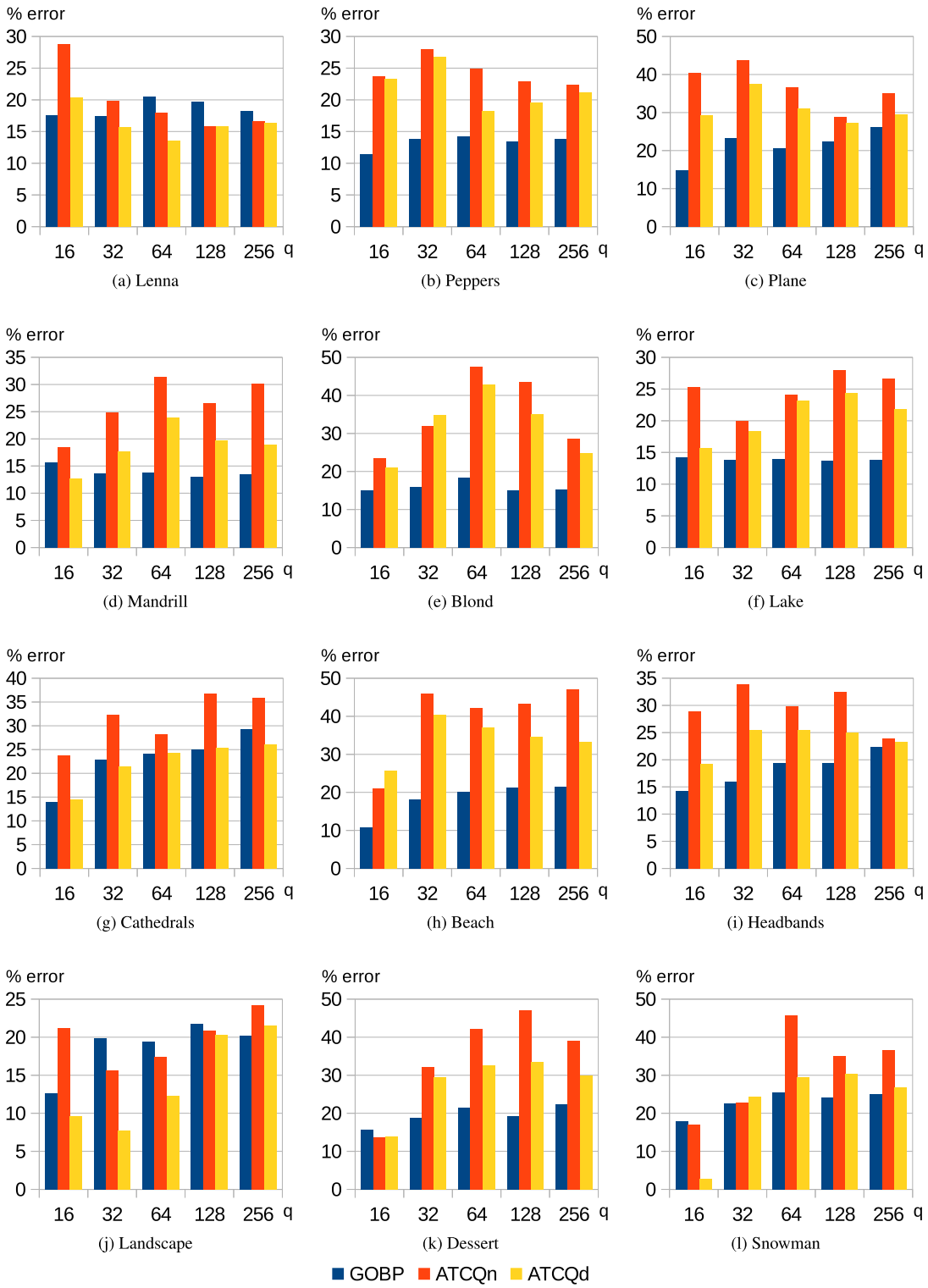


FIGURE 5. Percentage of error reduction obtained by WATCQd compared to GOBP, ATCQ without disconnection of ants (ATCQn) and ATCQ with disconnection of ants (ATCQd). (*q*: number of colors of the quantized palette.)

quantized image.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} ||p_i - p'_i||^2 \tag{12}$$

Five palette sizes were used in the tests: $q = \{16, 32, 64, 128, 256\}$. To apply the operations of the ATCQ algorithm, the values considered for the α parameter were those proposed in [23]: $\alpha = \{0.25, 0.30, 0.35, 0.40, 0.45, 0.50\}$.

TABLE 5. *MSE* and execution time (*T*) (milliseconds) for several color quantization methods: Variance-based method (VB), Median-cut (MC), Octree (OC), Binary splitting (BS), BS + ATCQ, Neuquant (NQ) and LBG. (*q*: Number of colors of the quantized palette.)

		VB		MC		OC		BS		BS+ATCQ		NQ		LBG	
	q	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T
Lenna	16	358.13	215	462.65	13	681.44	170	258.29	144	226.22	192	322.05	101	239.93	309
	32	203.07	251	425.85	13	482.03	174	138.44	178	123.82	263	152.13	152	128.81	501
	64	135.86	317	362.61	13	212.92	183	82.44	233	75.35	395	85.60	227	78.46	848
	128	94.68	358	276.33	13	140.53	182	53.04	284	48.10	595	53.73	345	50.42	1447
	256	69.41	418	193.25	14	74.12	187	35.28	367	31.59	929	34.88	567	35.42	2184
Peppers	16	751.90	247	1034.24	14	1593.03	171	531.57	153	468.58	196	556.86	106	456.79	294
	32	451.10	321	866.55	14	777.14	167	311.66	175	254.22	269	283.69	167	276.69	490
	64	304.21	395	771.03	14	495.51	172	173.75	223	145.66	486	166.37	273	152.75	836
	128	212.68	467	585.41	13	308.76	176	106.85	283	91.29	580	95.69	357	97.97	1525
	256	147.18	507	397.29	15	156.24	188	68.72	349	58.38	926	63.08	591	64.28	2579
Plane	16	187.84	193	509.91	13	395.23	169	174.72	130	141.22	173	311.72	96	173.45	280
	32	123.56	223	374.70	13	342.23	166	83.12	144	68.10	229	123.70	143	86.98	456
	64	80.73	275	237.93	13	225.98	170	46.22	199	39.30	349	57.57	207	45.68	740
	128	52.60	318	187.93	13	133.59	177	28.06	249	24.24	535	29.71	302	27.26	1278
	256	36.85	381	147.53	13	52.31	181	18.42	332	15.77	1045	21.09	566	18.01	1950
Mandrill	16	924.59	270	1292.35	15	1579.16	171	738.08	158	656.39	200	847.77	103	694.49	369
	32	531.03	342	984.05	16	1094.11	172	441.30	194	397.18	283	456.13	163	425.54	517
	64	346.58	426	881.29	17	576.19	175	286.93	238	248.82	413	272.25	242	262.97	761
	128	248.02	503	776.65	17	357.13	179	183.51	298	160.71	613	168.22	350	171.28	1166
	256	181.94	616	676.87	18	195.82	182	117.85	377	103.18	977	109.34	679	112.76	1934
Blond	16	425.38	128	549.85	12	502.45	176	212.36	144	171.18	180	265.53	95	243.08	269
	32	232.43	134	374.69	13	477.64	169	111.28	185	95.20	270	123.89	152	95.27	436
	64	129.43	164	217.55	13	163.08	178	63.07	227	51.89	399	66.32	222	53.50	752
	128	82.22	171	185.81	13	89.86	185	38.46	286	32.04	597	38.30	355	33.93	1336
	256	54.42	206	123.57	13	50.79	186	23.65	389	19.45	952	23.86	562	21.40	2418
Lake	16	507.32	292	1317.94	14	957.66	169	365.21	152	336.54	203	436.04	100	367.02	275
	32	357.26	379	1208.87	14	922.04	167	245.65	170	218.92	272	274.45	173	252.22	409
	64	251.70	471	984.62	15	466.14	170	162.89	224	140.32	389	164.26	238	169.43	643
	128	170.97	559	692.56	15	198.49	182	107.00	277	90.70	612	100.88	355	103.41	1280
	256	120.10	682	523.09	16	159.21	179	68.36	358	57.65	966	65.70	572	69.86	2043

To compare the results of the new method with those obtained by GOBP and ATCQ applied independently, both methods were applied to the same images. ATCQ was tested with the same α values previously indicated and both variants of this method were considered: the general algorithm that allows the disconnection of ants was labeled ATCQd, whereas the variant without disconnection of ants was labeled ATCQn. The proposed method was combined with both variants of ATCQ and the labels WATCQd and WATCQn were used to identify which variant of ATCQ was considered in each case. Tables 2 to 4 show the results of GOBP, ATCQ and WATCQ corresponding to 20 independent tests performed for each image and palette size. The tests were executed on a PC running Linux operating system, with an AMD Ryzen 7 1800X Turbo processor (4.0 GHz) and 8 GBytes of RAM. Since GOBP and WATCQn generate the same quantized image for each palette size in all the tests, only one error value is provided; for the other methods, the minimum *MSE*, the average value and the standard deviation are provided (Tables 2 and 3). The execution time reported in Table 4 is measured in milliseconds.

It is clearly observed that the proposed method always obtains better images than GOBP and ATCQ applied separately. Obviously, when the execution time is compared, it is larger for the proposed method. However, the improvement obtained in the image quality should be taken into account to determine if this increase in the execution time is worthwhile.

To better analyze the improvement obtained by the proposed method, Fig. 5 shows the percentage of error reduction obtained when comparing WATCQd to GOBP, ATCQn and ATCQd (only one variant of WATCQ is compared to simplify the graphic representation). Since the average *MSE* results of both variants of ATCQ are very large in some cases, this figure compares the minimum *MSE*. It is clearly observed that the improvement obtained is greater when WATCQd is compared to ATCQ than when compared to GOBP. The percentage of error reduction ranges in the interval [10.8%, 29.4%] when GOBP is compared, in [13.6%, 47.4%] when ATCQn is compared and in [2.8%, 42.8%] when ATCQd is compared. Although in the last case the lower limit of the interval is very small, only 3 values lower than 10% are obtained when WATCQd is compared to ATCQd. The analysis of the previous results indicates that the increase in execution time results in a significant improvement in the quality of the final image.

As indicated in the previous paragraph, the average *MSE* value of ATCQd and ATCQn is very large in some cases. This is because the α values used for the tests have not been

TABLE 6. *MSE* and execution time (*T*) (milliseconds) for several color quantization methods. Variance-based method (VB), Median-cut (MC), Octree (OC), Binary splitting (BS), BS + ATCQ, Neuquant (NQ) and LBG. (*q*: Number of colors of the quantized palette.)

		VB		MC		OC		BS		BS+ATCQ		NQ		LBG	
	q	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T	MSE	T
Cathedrals	16	222.35	216	375.78	12	1603.25	173	159.02	145	133.13	174	210.52	87	146.30	334
	32	105.84	267	284.29	12	316.46	175	72.42	164	64.48	259	93.72	136	66.00	507
	64	61.50	318	169.33	12	109.02	183	38.57	207	34.11	362	48.67	202	35.55	826
	128	40.15	370	142.12	15	69.79	183	22.43	245	19.69	561	24.33	375	21.85	1407
	256	26.16	424	99.17	13	45.83	190	13.29	319	11.88	887	15.28	565	13.65	2475
Beach	16	357.80	260	800.00	13	1164.78	179	358.98	143	219.73	208	501.48	101	287.01	323
	32	211.50	341	671.30	13	446.43	174	179.49	177	150.04	265	178.32	147	146.48	507
	64	123.49	421	557.04	13	309.23	180	93.22	210	79.09	366	91.86	216	82.12	850
	128	81.50	536	446.84	13	134.36	182	52.30	265	45.78	558	52.71	349	49.59	1426
	256	52.92	649	292.27	14	81.77	189	31.89	330	28.11	865	32.95	539	31.28	2581
Headbands	16	299.81	228	1517.76	13	909.26	171	245.61	152	213.98	206	360.14	95	232.36	265
	32	184.82	304	1195.51	13	430.86	176	149.52	186	124.51	284	188.99	160	126.61	445
	64	123.49	412	878.94	13	192.74	180	84.86	232	71.23	412	99.49	238	79.05	663
	128	79.90	474	519.00	14	128.22	182	51.30	288	43.06	629	52.12	361	45.67	1249
	256	53.29	440	314.55	14	53.17	189	31.28	363	25.76	1006	30.77	585	27.58	2284
Landscape	16	261.31	212	625.54	13	925.61	175	226.07	167	204.33	225	253.21	99	232.66	282
	32	164.35	238	419.37	13	576.99	174	129.18	208	107.72	305	139.03	166	109.57	556
	64	113.61	281	352.09	13	185.19	179	66.61	259	57.59	434	70.13	225	60.76	768
	128	84.46	335	257.61	13	149.46	182	36.48	313	32.69	659	37.17	365	36.92	1163
	256	61.90	380	185.22	13	53.17	191	22.18	400	19.87	1057	23.75	569	23.21	1953
Dessert	16	337.47	226	563.62	13	1096.19	170	283.86	159	259.94	210	349.16	85	277.04	288
	32	191.72	262	451.64	13	426.95	170	152.16	189	133.40	289	176.26	143	147.28	465
	64	118.69	315	311.91	13	203.65	175	85.29	233	70.66	414	90.70	205	78.13	786
	128	82.38	383	233.09	13	118.32	182	48.50	289	41.77	624	51.09	316	47.18	1381
	256	56.92	441	172.45	14	67.32	185	29.42	372	24.83	1014	30.21	539	28.65	2514
Snowman	16	349.10	231	452.85	13	719.26	168	289.76	156	245.36	206	388.00	94	291.60	259
	32	216.35	295	366.86	13	559.57	171	163.63	192	134.07	285	202.14	152	161.02	445
	64	118.13	383	305.82	13	334.53	175	87.54	245	68.77	428	90.05	219	73.35	878
	128	70.03	466	212.60	14	134.31	181	46.97	300	38.09	641	44.23	349	44.98	1483
	256	43.68	541	165.97	14	84.45	183	27.48	395	22.33	1045	27.51	601	27.21	2558

adjusted based on the palette size. The effect is especially evident for the Mandrill image, with a very large average error and standard deviation because the smallest α values generate quantized palettes with very few colors (only 6 colors in the worst case). As described in [23], the best result of ATCQ is obtained, in general, for the lowest α value that generates a palette with Q_{max} colors. Such article also shows that the α value should be larger as the palette size increases, in order to obtain quantized images with the number of desired colors. This recommendation has not been taken into consideration in the tests performed, since the same α set has been used to test WATCQ and ATCQ for all the palette sizes. The average error and the standard deviation obtained for both methods indicate that the α parameter has little influence on WATCQ; for this reason, the same set of α values has been used for all the palette sizes.

The execution time of GOBP is independent of the palette size, but this is not true for ATCQ. Therefore, the execution time of WATCQ is also influenced by the palette size. Obviously, WATCQn is faster than WATCQd because ATCQn is faster than ATCQd. Nevertheless, it is observed that the execution time of WATCQ is not approximately equal to the sum of the execution time of GOBP and ATCQ methods, as might be expected. This is due to the effect of the α value in ATCQ. As previously discussed, when a small value is considered for the α parameter, the final palette obtained by ATCQ includes less than Q_{max} colors. Nevertheless, WATCQ always generates a final palette that includes q colors. Moreover, ATCQ progressively includes the subtrees as the iterations progress, but when ATCQ is applied in WATCQ the initial tree always includes q subtrees. Therefore, WATCQ consumes more time than ATCQ to select the best subtree for an ant, since in the first case more subtrees must be analyzed.

When comparing both variants of the proposed method, the best option is WATCQn, because it consumes less time than WATCQd and can generate quantized images with similar quality. WATCQd consumes a maximum of 846 milliseconds to generate an image with 256 colors, and this value reduces to 546 milliseconds when WATCQn is considered. It is also observed that the differences in execution time are greater for the smallest palettes.

To complement the results presented in the article, the quantized images obtained by GOBP, ATCQ and WATCQ for some of the test images are included as supplementary material.

Some of the color quantization methods described in Section III were applied to the test images, in order to compare their results with those obtained by WATCQ.

TABLE 7. Results for several color quantization methods: K-means (KM), PSO, ATCQ + FA and ITATCQ. (q: Number of colors of the quantized palette; MSE_m: Minimum MSE; MSE_a: Average MSE; dev: Standard deviation of MSE; T: Execution time (milliseconds).)

		KM				PSO				ATCQ+FA				ITATCQ			
	q	MSE_m	MSE_a	dev	T	MSE_m	MSE_a	dev	T	MSE_m	MSE_a	dev	T	MSE_m	MSE_a	dev	T
Lenna	16	227.35	247.91	14.2	830	211.01	216.41	4.5	15049	251.85	253.70	1.5	1067	257.05	293.99	21.2	179
	32	126.60	140.45	8.7	1604	121.84	124.17	2.7	28770	139.72	140.92	0.8	1680	143.03	167.46	27.6	277
	64	77.55	84.91	3.4	3048	75.56	76.72	1.2	55587	80.29	81.10	0.5	2671	82.23	96.53	15.1	495
	128	50.79	52.90	1.6	5996	49.21	50.47	1.0	105978	50.68	51.62	0.5	4074	51.45	56.13	6.5	866
	256	33.93	34.86	0.7	11864	32.21	33.00	0.5	223187	33.04	33.39	0.2	5989	33.04	34.18	1.9	1563
Peppers	16	419.07	462.00	21.0	817	404.31	417.91	6.0	15574	501.35	503.33	1.5	958	508.56	1137.89	797.5	162
	32	258.20	282.09	17.3	1553	231.50	236.70	3.2	31922	291.78	294.61	1.4	1602	294.43	362.83	84.2	275
	64	141.82	160.80	7.9	3016	136.91	140.77	2.2	60356	159.97	161.36	0.8	2718	160.88	181.35	20.7	472
	128	90.90	99.39	5.1	6332	86.93	88.46	1.4	124997	97.05	98.27	0.6	4144	97.31	103.71	8.3	877
	256	60.47	63.03	2.6	11860	57.61	58.29	0.7	231727	62.59	63.38	0.5	5972	62.55	64.63	1.9	1621
Plane	16	201.91	283.79	67.8	825	162.27	176.64	10.3	15795	160.79	164.73	2.1	1009	166.94	201.91	34.7	164
	32	128.02	184.21	58.9	1602	72.90	104.40	19.0	30094	100.24	101.27	0.7	1573	100.35	121.31	21.5	268
	64	48.05	94.89	27.2	3015	46.43	52.36	5.1	64354	47.26	47.99	0.3	2565	47.44	65.51	18.9	474
	128	39.29	47.02	10.5	5978	29.73	33.60	1.8	120023	27.36	27.89	0.3	3637	27.34	38.87	15.8	855
	256	24.43	29.42	6.9	12153	19.99	22.29	1.2	248386	17.93	18.20	0.1	4862	17.58	18.12	0.8	1527
Mandrill	16	655.16	709.48	51.9	820	632.37	638.89	5.0	14655	713.18	717.87	2.1	1027	710.27	737.61	27.1	189
	32	397.44	412.12	13.0	1602	378.81	382.53	2.3	27590	405.78	409.85	1.8	1547	409.44	442.71	20.7	290
	64	253.28	259.23	5.3	3173	240.57	242.03	1.0	55524	260.22	270.12	4.9	2418	280.69	308.99	20.1	503
	128	161.49	164.24	1.8	5941	154.67	156.22	1.0	106722	168.67	171.72	1.8	3592	175.17	189.85	18.1	878
	256	103.60	106.64	1.7	12219	99.84	100.84	0.8	220198	111.98	112.52	0.3	5252	114.14	118.38	4.8	1601
Blond	16	202.08	262.10	45.6	817	158.08	167.67	11.5	15962	191.85	194.43	1.5	1066	197.48	231.60	32.1	172
	32	113.74	149.64	25.5	1631	85.49	87.13	1.4	30336	101.88	103.96	1.5	1684	106.44	139.98	32.5	281
	64	55.92	84.61	15.3	3152	51.48	52.53	0.6	62289	59.23	60.20	1.1	2698	68.89	82.24	17.8	478
	128	34.48	50.03	14.1	6194	31.34	32.78	1.0	130423	37.30	38.30	0.5	4148	38.88	40.91	1.5	861
	256	22.69	26.21	5.5	12000	20.07	20.97	0.7	248912	21.82	22.25	0.2	5727	21.68	22.85	36.5	1636
Lake	16	332.02	371.94	32.5	821	315.25	322.39	3.7	15292	355.41	358.24	1.4	1006	365.52	698.08	261.1	169
	32	220.68	235.44	10.1	1569	206.08	207.87	1.5	29006	221.81	223.89	1.0	1658	229.40	362.30	207.0	278
	64	144.43	155.96	8.0	3078	133.77	135.72	1.6	55110	155.04	155.95	0.7	2779	156.88	173.74	15.5	485
	128	95.08	101.34	3.8	6087	88.94	90.21	0.8	112275	102.39	103.00	0.3	4328	103.32	107.64	6.0	862
	256	63.25	66.07	1.9	11946	58.46	59.63	0.8	137699	63.29	63.94	0.3	6439	64.36	65.74	1.4	1598

The following methods were considered: Variance-based method [21], Median-cut [19], Octree [18], Binary splitting [20], BS + ATCQ [33], Neuquant [22], LBG [57], K-means [24], Particle swarm optimization (PSO) [46], ATCQ + FA [32], and ITATCQ [31]. LBG is based on the vector quantization method proposed in [57], which has also been applied to color quantization. This method applies the same operations as K-means to define the clusters, but it uses a splitting process to obtain the initial centroids. Some of these methods use parameters whose values were defined as follows. The sampling factor of Neuquant was set to 1, since this value allows the method to generate the best quantized images. As will be shown in the results of the tests, K-means and PSO consume much more time than the other methods. For this reason, only 5 iterations of these methods were executed. To perform a fair comparison, the same number of iterations was considered for the other iterative methods (ITATCQ and ATCQ + FA). The values used for the other PSO parameters were: cognitive and social parameters equal to 1.49, inertia equal to 0.72, range for the velocity of the particles: $[-5, 5]$ and probability of application of K-means equal to 0.1 (as proposed in [46]), 5 particles and 5 iterations of K-means when this method must be applied to a particle. The three methods based on ATCQ (BS + ATCQ, ATCQ + FF and ITATCQ), were applied considering the ATCQ variant without disconnection of ants, since it consumes less time. In addition, BS + ATCQ and ITATCQ were executed with the six α values proposed to test WATCQ. ATCQ + FA was executed considering the same number of individuals as PSO (5 fireflies) and five α values were selected: $\{0.25, 0.30, 0.35, 0.40, 0.45\}$. The results of 20 independent tests are reported in Tables 5 to 8.

WATCQ clearly obtains better images than all the methods reported in Tables 5 and 6, except BS + ATCQ. The improvement is more evident for the Median-cut and Octree methods, whereas the smallest differences appear when WATCQ is compared to Binary splitting and LBG. As the palette size increases, the differences in the *MSE* error also increase when WATCQ is compared to Median-cut and Variance-based methods; however, for Neuquant the opposite occurs. This can be clearly observed in Fig. 6, which shows the percentage of error reduction obtained when the average *MSE* of WATCQd is compared to the error obtained by Variance-based, Median-cut, Octree, Binary splitting and Neuquant methods. When the execution time is compared, the results of the analysis are different for both variants of the proposed method. WATCQn consumes less time than the five methods in many cases, but WATCQd consumes more time.

TABLE 8. Results for several color quantization methods. K-means (KM), PSO, ATCQ + FA and ITATCQ. (q: Number of colors of the quantized palette; MSE_m: Minimum MSE; MSE_a: Average MSE; dev: Standard deviation of MSE; T: Execution time (milliseconds).)

		KM				PSO				ATCQ+FA				ITATCQ			
	q	MSE_m	MSE_a	dev	T	MSE_m	MSE_a	dev	T	MSE_m	MSE_a	dev	T	MSE_m	MSE_a	dev	T
Cathedrals	16	143.17	176.69	30.1	845	124.34	127.93	2.4	14726	135.96	137.90	1.4	986	140.80	314.80	157.7	171
	32	69.89	84.05	12.0	1597	62.87	64.34	1.0	29812	75.65	77.47	1.3	1592	77.15	167.84	132.4	268
	64	39.63	44.86	2.6	3102	35.19	36.66	1.0	56207	40.40	41.41	0.5	2578	40.22	101.02	117.2	471
	128	24.13	26.72	1.8	6106	21.74	22.24	0.5	115725	22.99	23.87	0.7	3826	23.96	27.50	4.7	810
	256	15.90	16.88	0.7	12393	13.89	14.76	0.4	220688	14.00	14.43	0.2	5416	14.18	14.75	0.6	1479
Beach	16	292.16	324.05	24.3	845	270.75	274.90	4.8	14867	335.75	338.63	1.8	1022	335.66	746.50	272.8	168
	32	149.28	165.65	14.8	1630	140.91	146.27	4.4	29037	186.49	188.89	1.7	1584	193.57	424.36	244.9	270
	64	82.94	91.01	5.8	3169	76.43	80.06	2.4	57533	102.69	105.55	1.6	2733	108.31	285.59	230.8	457
	128	48.77	52.15	2.4	6034	47.74	48.57	0.7	112493	56.14	57.15	0.7	4215	57.68	149.51	139.3	843
	256	30.89	32.97	1.4	12249	29.91	30.81	0.7	204101	33.40	34.01	0.3	6139	34.00	47.91	15.4	1548
Headbands	16	235.52	302.86	69.0	884	201.19	205.47	3.1	15540	238.84	240.10	0.9	1045	237.73	391.24	148.4	172
	32	142.46	180.87	63.1	1600	118.06	123.72	3.5	30664	136.55	138.78	1.4	1650	139.48	220.65	106.1	278
	64	81.35	108.19	10.2	3083	73.77	76.78	2.4	59336	85.64	86.50	0.5	2729	85.16	123.58	44.6	491
	128	57.26	65.05	6.8	6082	44.27	48.38	2.4	115355	48.08	48.68	0.3	4066	48.44	64.41	25.6	874
	256	33.40	38.86	2.9	12401	29.52	30.33	0.6	220748	27.95	28.31	0.1	5814	27.39	30.72	4.2	1558
Landscape	16	191.02	220.75	16.3	864	189.04	195.20	4.1	15087	216.48	219.96	1.2	1015	220.24	451.90	251.8	170
	32	110.46	124.51	11.1	1626	99.60	104.72	3.7	31439	110.51	111.94	0.9	1575	110.25	230.25	171.0	283
	64	61.36	72.95	8.2	3215	56.95	59.33	1.5	55458	61.96	62.89	0.4	2630	61.80	86.15	42.7	490
	128	37.28	42.14	3.0	6220	35.54	36.74	0.9	116756	36.83	37.24	0.3	3918	36.80	41.03	6.6	922
	256	24.22	25.85	0.8	12206	22.38	23.27	0.5	230704	22.92	23.26	0.2	5095	22.85	22.94	28.6	1638
Dessert	16	245.29	313.21	47.0	833	241.24	246.93	5.1	15365	255.11	259.81	1.8	1002	260.38	335.42	61.5	168
	32	131.54	154.34	18.4	1603	121.23	125.41	2.2	28863	142.15	144.25	1.0	1597	145.27	197.91	43.1	279
	64	77.93	85.53	5.2	3090	69.39	70.96	1.5	55837	80.65	82.38	1.0	2722	81.30	100.79	16.6	497
	128	47.00	52.05	2.9	6205	42.85	43.84	1.1	106298	51.89	52.62	0.4	4194	51.92	57.54	4.0	884
	256	29.81	32.12	1.4	12381	25.82	26.81	0.5	222618	30.31	31.10	0.6	6182	30.39	32.48	2.1	1578
Snowman	16	256.49	302.35	32.7	848	229.52	238.17	7.8	14579	252.38	255.58	1.7	1022	264.72	383.22	133.7	171
	32	133.30	174.26	27.2	1644	118.33	124.55	5.2	29816	135.80	137.54	0.8	1664	136.09	163.84	18.2	277
	64	74.72	89.74	10.5	3148	65.07	69.63	3.0	57066	66.56	67.10	0.4	2687	68.28	95.21	22.6	482
	128	43.76	50.66	2.7	6186	39.92	40.83	0.5	110831	41.65	42.15	0.2	4270	41.76	52.36	16.4	932
	256	27.73	30.85	2.4	12368	25.57	25.51	0.6	223735	25.19	25.67	0.4	6414	24.79	26.69	2.3	1632

WATCQn always consumes less time than Neuquant; on the contrary, it always consumes more time than Median-cut. Moreover, WATCQn is faster than Binary splitting, Octree and Variance-based methods for all the images when $q \leq 64$, and it is also faster than Binary splitting and Variance-based methods for some cases (8 and 15 cases out of 24, respectively) when $q > 64$. On the other hand, WATCQd always consumes more time than Median-cut, Neuquant, Octree and Binary splitting. Nevertheless, it consumes less time than the Variance-based method for 27 out of 60 cases, corresponding to the palettes with fewer colors.

LBG consumes more time than the two variants of WATCQ and obtains worse images than the new method in almost all the cases (except in 3 cases). The percentage of error reduction obtained by LBG for these three cases is 3.4% at most. On the contrary, the reduction of the error obtained by WATCQd for the other 57 cases varies in [0.7%, 29.3%] and exceeds 5% for 47 cases. The percentages are shown in Fig. 7, which shows the percentage of error reduction obtained when the average *MSE* of WATCQd is compared to the error obtained by other methods.

ATCQ + FA and ITATCQ consume more time than WATCQ, even though only 5 iterations of these methods were executed. In addition, the *MSE* results of both methods are always improved by WATCQ, as can be observed in Fig. 7.

It is also observed that the *MSE* results of BS + ATCQ are similar to those of WATCQ, although the new method consumes less time. WATCQ obtains better images for 30 out of 60 cases. For the other cases, the percentage of error reduction obtained by BS + ATCQ compared to WATCQd ranges in [0.2%, 7.3%], although the percentage exceeds 4% for only 5 cases. WATCQn consumes less time than BS + ATCQ for all the palette sizes; the percentage of time reduction obtained in this case varies in between 39% and 73%. On the other hand, WATCQd consumes more time than BS + ATCQ only in 12 cases corresponding to small palette sizes.

When comparing PSO and WATCQ, the first aspect to highlight is that the execution time of both methods is very different. Although PSO can obtain images with better average *MSE* than WATCQ for 28 out of 60 cases, the percentage of error reduction is 8.2% at most and only in 5 cases this value exceed 5%. Most of the remaining 32 cases improved by WATCQ correspond to palettes with 128 and 256 colors. It can also be observed that WATCQ obtains better results than PSO for all the palettes when the Plane image is considered, with a percentage of error reduction that exceeds 22% in all the cases. Therefore, although PSO can obtain better images than WATCQ for the smallest palettes, the improvement is very small and requires a lot of computing time, which can reduce the utility of such improvement.

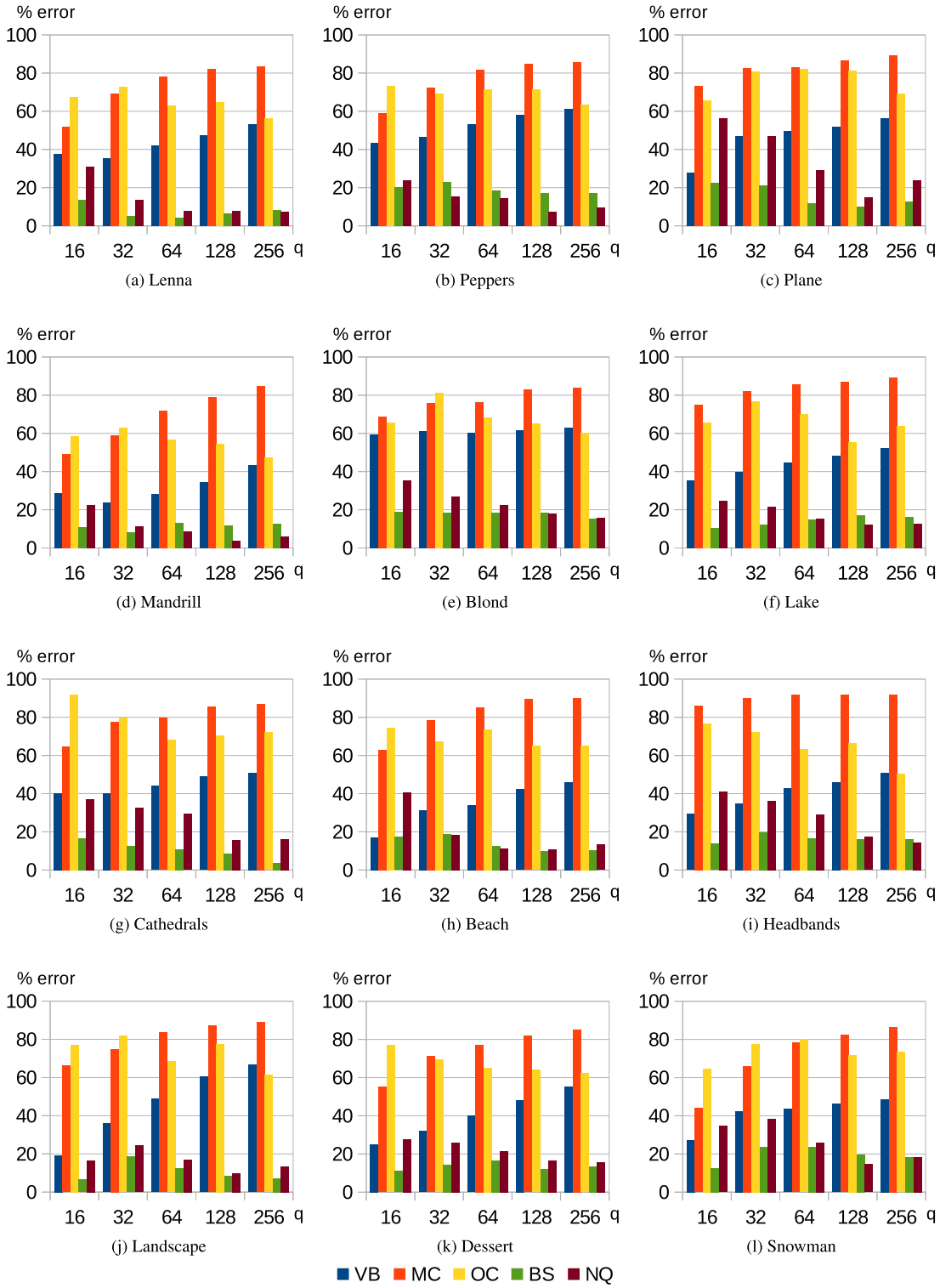


FIGURE 6. Percentage of error reduction obtained by WATCQd compared to Variance-based (VB), Median-cut (MC), Octree (OC), Binary splitting (BS) and Neuquant (NQ). (*q*: Number of colors of the quantized palette.)

When K-means results are analyzed, it is observed that WATCQ obtains better average errors for all the cases. When comparing the minimum *MSE* errors, K-means obtains better values than both variants of WATCQ for only 10 out of 60 cases, corresponding mainly to palettes with 16 and 32 colors. On the other hand, it is observed that the execution time of K-means is much greater than that of WATCQ, even though only 5 iterations of K-means have been executed.

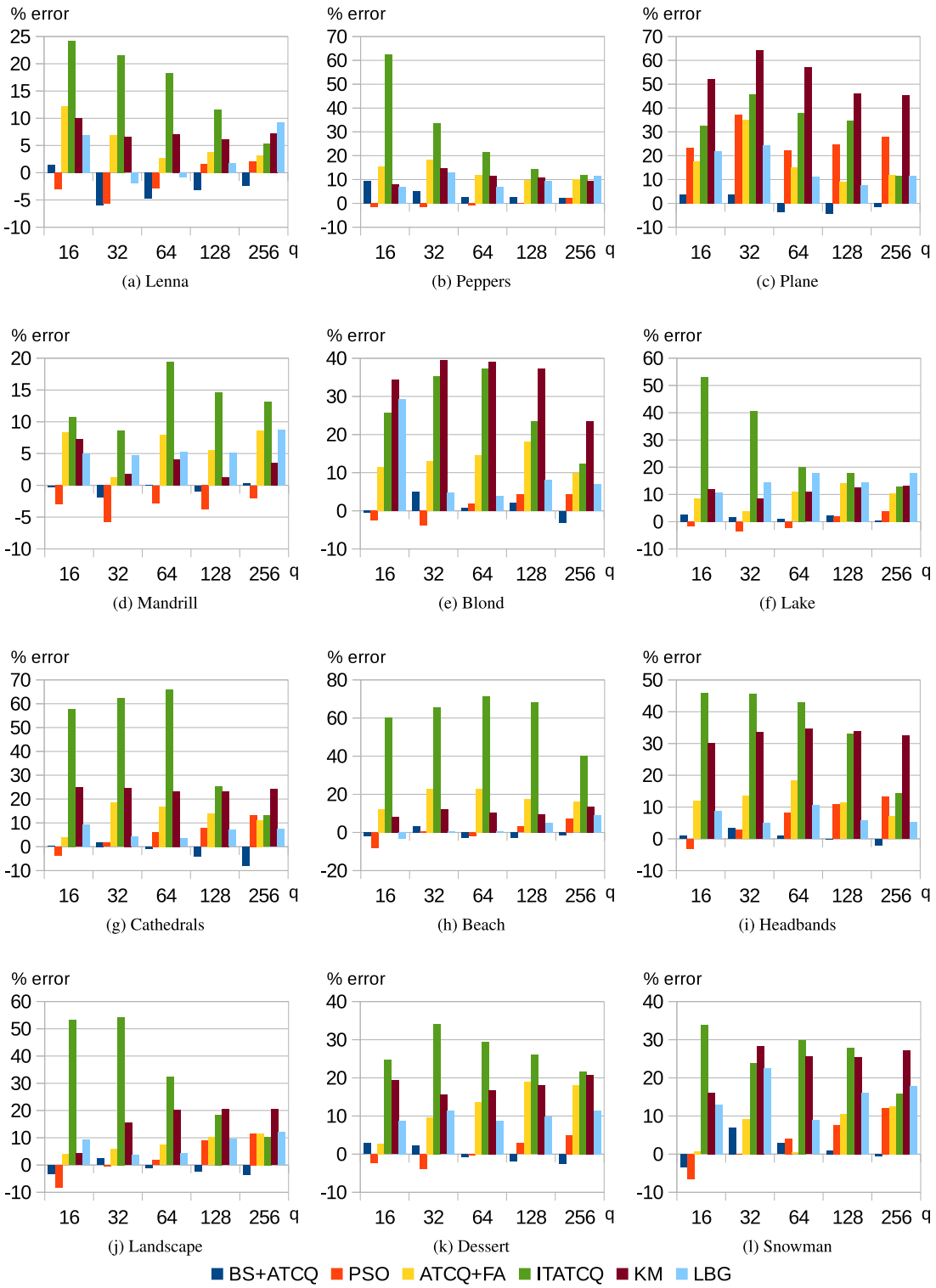


FIGURE 7. Percentage of error reduction obtained by WATCQd compared to BS + ATCQ, PSO, ATCQ + FA, ITATCQ, K-means (KM) and LBG. (*q*: Number of colors of the quantized palette.)

The K-means method can be applied to random initial centroids, as was done in the tests reported in Tables 7 and 8, but it can also be applied to selected centroids. For this reason, new tests were performed taking as initial centroids the values of the palette generated by GOBP. Table 9 shows the results of these tests at iterations 1, 3, and 5. This table clearly shows that iteration 1 of K-means generates worse results than both variants of WATCQ: K-means is slower and

TABLE 9. *MSE* and execution time (milliseconds) of K-means applied to the palette generated by GOBP. (q: Number of colors of the quantized palette.)

		iter. 1		iter. 3		iter. 5				iter. 1		iter. 3		iter. 5	
	q	MSE	T	MSE	T	MSE	T		q	MSE	T	MSE	T	MSE	T
Lenna	16	226.03	233	217.06	570	213.80	898	Cathedrals	16	133.81	237	130.83	567	129.49	904
	32	132.29	377	127.74	1013	125.69	1643		32	64.13	413	62.16	1076	61.57	1726
	64	80.25	675	76.33	1936	75.04	3169		64	35.56	716	33.99	2018	33.53	3300
	128	50.55	1308	48.89	3731	48.44	6193		128	21.35	1382	20.18	3909	19.82	6409
	256	33.24	2525	32.08	7379	31.68	12170		256	13.79	2594	12.92	7590	12.63	12530
Peppers	16	426.01	233	413.34	557	409.19	905	Beach	16	298.51	239	288.68	572	276.69	932
	32	241.41	407	235.54	1055	233.90	1684		32	146.20	396	142.05	1063	140.84	1726
	64	142.99	700	140.05	1943	138.99	3217		64	82.53	704	78.53	1982	76.86	3244
	128	89.54	1338	87.39	3804	86.50	6233		128	48.07	1359	45.82	3866	44.99	6351
	256	57.99	2532	56.13	7323	55.41	12128		256	29.48	2574	28.19	7588	27.75	12507
Plane	16	135.94	233	132.23	561	130.64	890	Headbands	16	213.73	239	206.56	585	205.09	937
	32	66.61	382	65.12	1026	64.45	1654		32	121.72	398	117.58	1055	116.24	1707
	64	41.39	695	39.85	1942	39.26	3177		64	71.78	717	68.55	1995	67.50	3267
	128	26.45	1302	25.11	3760	24.63	6200		128	44.21	1336	42.20	3883	41.44	6436
	256	17.35	2503	16.26	7380	15.97	12187		256	27.52	2620	25.69	7626	25.16	12611
Mandrill	16	658.42	237	643.30	566	638.24	903	Landscape	16	211.75	240	207.94	568	205.26	905
	32	406.55	403	394.13	1057	388.18	1690		32	105.98	398	101.52	1084	100.24	1741
	64	249.90	687	243.61	1947	241.00	3179		64	59.28	710	57.54	1988	56.74	3259
	128	163.51	1297	158.64	3704	156.73	6135		128	34.65	1368	33.03	3883	32.47	6408
	256	103.73	2512	101.19	7355	99.98	12119		256	21.40	2611	20.54	7634	20.30	12579
Blond	16	173.64	235	169.09	580	166.64	927	Dessert	16	252.86	237	246.20	571	243.37	940
	32	91.52	400	87.36	1043	86.57	1692		32	131.47	389	125.81	1097	124.47	1807
	64	52.63	716	50.57	1967	49.96	3223		64	72.34	710	69.54	2016	68.67	3294
	128	32.21	1420	31.07	4039	30.72	6631		128	43.54	1349	41.19	3872	40.38	6395
	256	20.91	2586	19.91	7461	19.55	12303		256	26.41	2651	24.88	7678	24.34	12607
Lake	16	327.83	240	319.52	569	317.44	900	Snowman	16	254.60	243	244.26	594	236.17	941
	32	216.59	391	209.92	1040	207.74	1682		32	124.97	409	119.56	1064	118.34	1715
	64	140.19	734	136.08	2066	134.78	3392		64	67.66	719	64.38	1989	63.42	3299
	128	89.58	1310	87.28	3790	86.48	6241		128	38.75	1329	36.70	3852	35.78	6363
	256	58.35	2513	56.67	7392	55.88	12192		256	23.50	2601	22.06	7560	21.59	12483

generates worse images in almost all the cases (except in 3 cases when WATCQn is compared). In contrast, the results of iterations 3 and 5 correspond to better images that those obtained by WATCQ in almost all the cases. The reduction in the error obtained by the results of K-means at iteration 5 is 7% at most and exceeds 5% for very few cases (7 and 9 cases when compared to WATCQd and WATCQn, respectively). Therefore, although K-means applied to the results of GOBP can generate images a little better than WATCQ, it also requires much more computational effort. The execution time of K-means is only comparable to that of WATCQd for images with 16 colors if iteration 1 is considered, but in this case K-means always generates worse images than WATCQd.

As a summary of the previous analysis, it can be concluded that WATCQ can obtain better results than the other methods analyzed, since it obtains better images than most of these methods and only obtains worse images when compared with much slower methods.

VIII. CONCLUSION

This article presents a color quantization method, called WATCQ, that combines the operations of GOBP and ATCQ methods in order to improve the quality of the resulting image but with low computational cost. GOBP is a well-known splitting method that can generate a quantized image quickly, while ATCQ is a recently proposed clustering-based method that can generate better images than some other color quantization techniques.

GOBP is a deterministic method that always generates the same quantized image for a given palette size, whereas the parameters of ATCQ allow it to generate several quantized images for each palette size. For this reason, the proposed method uses the result of GOBP as a starting point to apply ATCQ. Two variants of ATCQ can be considered, depending on whether the disconnection of ants is allowed or not. Therefore, two variants of WATCQ have been considered, labeled WATCQd and WATCQn, each of which uses a variant of ATCQ.

Computational experiments show that the combined method always improves the quality of the quantized images generated by GOBP and ATCQ methods applied separately. In addition, the execution time is less than 846 milliseconds for all the images and palette sizes analyzed, and it is reduced to approximately 60 milliseconds when the palette size is 16. It is also observed that WATCQn is faster than WATCQd and the quality of the quantized images obtained by both variants is very similar. When WATCQn is considered, the execution time reduces to 537 milliseconds for the largest palette.

The proposed method has also been compared to other color quantization methods, generating better images than

some well-known techniques, such as Binary splitting, Neuquant, Octree, Median-cut and Variance-based methods. Moreover, in many cases WATCQn consumes less time than these methods for the smallest palette sizes, except when comparing Median-cut. Although Median-cut is very fast compared to WATCQ, it generates images much worse than WATCQ. Other clustering-based methods analyzed can generate better images than WATCQ in some cases, but the execution time required to obtain such images makes these methods not competitive with WATCQ.

Since the proposed method is fast, future research lines related to this method include real-time image processing applications. For example, these applications include the quality management systems for food production chains, where the analysis of food images must be carried out quickly. Certainly, this is the first practical application of the proposed method that we are currently testing.

REFERENCES

[1] X. Chen, S. Kwong, and J.-F. Feng, "A new compression scheme for color-quantized images," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 12, no. 10, pp. 904–908, Oct. 2002.

[2] W. Ding, Y. Lu, and F. Wu, "Enable efficient compound image compression in H.264/AVC intra coding," in *Proc. IEEE Int. Conf. Image Process.*, San Antonio, TX, USA, vol. 2, Sep./Oct. 2007, pp. 337–340.

[3] S. Jeong, C. S. Won, and R. M. Gray, "Image retrieval using color histograms generated by Gauss mixture vector quantization," *Comput. Vis. Image Understand.*, vol. 94, nos. 1–3, pp. 44–66, Apr./Jun. 2004.

[4] M. Singha and K. Hemachandran, "Content based image retrieval using color and texture," *Signal Image Process.*, vol. 3, no. 1, p. 39, Feb. 2012.

[5] G. H. Liu and J. Y. Yang, "Content-based image retrieval using color difference histogram," *Pattern Recognit.*, vol. 46, no. 1, pp. 188–198, 2013.

[6] O. Sertel, J. Kong, G. Lozanski, A. Shana'ah, U. Catalyurek, J. Saltz, and M. Gurcan, "Texture classification using nonlinear color quantization: Application to histopathological image analysis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process., (ICASSP)*, Las Vegas, NV, USA, Mar./Apr. 2008, pp. 597–600.

[7] O. Losson and L. Macaire, "CFA local binary patterns for fast illuminant-invariant color texture classification," *J. Real-Time Image Process.*, vol. 10, no. 2, pp. 387–401, Jun. 2015.

[8] M. Ponti, T. S. Nazaré, and G. S. Thumé, "Image quantization as a dimensionality reduction procedure in color and texture feature extraction," *Neurocomputing*, vol. 173, pp. 385–396, Jan. 2016.

[9] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 800–810, Aug. 2001.

[10] S. L. Phung, A. Bouzerdoum, and D. S. Chai, "Skin segmentation using color pixel classification: Analysis and comparison," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 1, pp. 148–154, Jan. 2005.

[11] Z. Yu, O. C. Au, R. Zou, W. Yu, and J. Tian, "An adaptive unsupervised approach toward pixel clustering and color image segmentation," *Pattern Recognit.*, vol. 43, no. 5, pp. 1889–1906, May 2010.

[12] T. D. Nguyen and G. Lee, "Color image segmentation using tensor voting based color clustering," *Pattern Recognit. Lett.*, vol. 33, no. 5, pp. 605–614, Apr. 2012.

[13] N.-Y. An and C.-M. Pun, "Color image segmentation using adaptive color quantization and multiresolution texture characterization," *Signal Image Video Process.*, vol. 8, no. 5, pp. 943–954, Jul. 2014.

[14] P. Tsai, Y.-C. Hu, and C.-C. Chang, "A color image watermarking scheme based on color quantization," *Signal Process.*, vol. 84, no. 1, pp. 95–106, Jan. 2004.

[15] C.-T. Kuo and S.-C. Cheng, "Fusion of color edge detection and color quantization for color image watermarking using principal axes analysis," *Pattern Recognit.*, vol. 40, no. 12, pp. 3691–3704, Dec. 2007.

[16] M. Garey, D. Johnson, and H. Witsenhausen, "The complexity of the generalized Lloyd–Max problem (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 255–256, Mar. 1982.

[17] X. Wu, "Efficient statistical computations for optimal color quantization," in *Graphics Gems II*, J. Arvo, Ed. New York, NY, USA: Academic, 1991, pp. 126–133.

[18] M. Gervautz and W. Purgathofer, "A simple method for color quantization: Octree quantization," in *Graphics Gems*, A. S. Glassner, Ed. San Diego, CA, USA: Academic, 1990, pp. 287–293.

[19] P. Heckbert, "Color image quantization for frame buffer display," in *Proc. 9th Annu. Conf. Comput. Graph. Interact. Tech.*, New York, NY, USA, 1982, pp. 297–307.

[20] M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Trans. Signal Process.*, vol. 39, no. 12, pp. 2677–2690, Dec. 1991.

[21] S. J. Wan, P. Prusinkiewicz, and S. K. M. Wong, "Variance-based color image quantization for frame buffer display," *Color Res. Appl.*, vol. 15, no. 1, pp. 52–58, Feb. 1990.

[22] A. H. Dekker, "Kohonen neural networks for optimal colour quantization," *Netw., Comput. Neural Syst.*, vol. 5, no. 3, pp. 351–367, 1994.

[23] M.-L. Pérez-Delgado, "Colour quantization with ant-tree," *Appl. Soft Comput.*, vol. 36, pp. 656–669, Nov. 2015.

[24] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[26] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Eng. Fac., Dept. Comput. Eng., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR-06, 2005.

[27] M. E. Celebi, "Improving the performance of k-means for color quantization," *Image Vis. Comput.*, vol. 29, no. 4, pp. 260–271, Mar. 2011.

[28] Q. Wen and M. E. Celebi, "Hard versus fuzzy c-means clustering for color quantization," *EURASIP J. Adv. Signal Process.*, vol. 2011, no. 1, pp. 1–12, Dec. 2011.

[29] M. E. Celebi, S. Hwang, and Q. Wen, "Colour quantisation using the adaptive distributing units algorithm," *Imag. Sci. J.*, vol. 62, no. 2, pp. 80–91, 2014.

[30] M. E. Celebi, Q. Wen, and S. Hwang, "An effective real-time color quantization method based on divisive hierarchical clustering," *J. Real-Time Image Process.*, vol. 10, no. 2, pp. 329–344, 2015.

[31] M.-L. Pérez-Delgado, "An iterative method to improve the results of ant-tree algorithm applied to colour quantisation," *Int. J. Bio-Inspired Comput.*, vol. 12, no. 2, pp. 87–114, 2018.

[32] M.-L. Pérez-Delgado, "Artificial ants and fireflies can perform colour quantisation," *Appl. Soft Comput.*, vol. 73, pp. 153–177, Dec. 2018.

[33] M.-L. Pérez-Delgado and J.-Á. R. Gallego, "A two-stage method to improve the quality of quantized images," *J. Real-Time Image Proc.*, pp. 1–25, 2018. doi: 10.1007/s11554-018-0814-8.

[34] H. J. Park, K. B. Kim, and E.-Y. Cha, "An effective color quantization method using octree-based self-organizing maps," *Comput. Intell. Neurosci.*, vol. 2016, Nov. 2016, Art. no. 5302957.

[35] Y. Ueda, T. Koga, N. Suetake, and E. Uchino, "Color quantization method based on principal component analysis and linear discriminant analysis for palette-based image generation," *Opt. Rev.*, vol. 24, no. 6, pp. 741–756, Dec. 2017.

[36] P. Scheunders, "A genetic c-means clustering algorithm applied to color image quantization," *Pattern Recognit.*, vol. 30, no. 6, pp. 859–866, Jun. 1997.

[37] Y.-C. Hu and B.-H. Su, "Accelerated k-means clustering algorithm for colour image quantization," *Imag. Sci. J.*, vol. 56, no. 1, pp. 29–40, 2008.

[38] H. Kasuga, H. Yamamoto, and M. Okamoto, "Color quantization using the fast k-means algorithm," *Syst. Comput. Jpn.*, vol. 31, no. 8, pp. 33–40, 2000.

[39] X. D. Yue, D. Q. Miao, L. B. Cao, Q. Wu, and Y. F. Chen, "An efficient color quantization based on generic roughness measure," *Pattern Recognit.*, vol. 47, no. 4, pp. 1777–1789, Apr. 2014.

[40] A. Khaled, R. F. Abdel-Kader, and M. S. Yasein, "A hybrid color image quantization algorithm based on k-means and harmony search algorithms," *Appl. Artif. Intell.*, vol. 30, no. 4, pp. 331–351, May 2016.

[41] D. Özdemir and L. Akarun, "A fuzzy algorithm for color quantization of images," *Pattern Recognit.*, vol. 35, no. 8, pp. 1785–1791, Aug. 2002.

[42] G. Schaefer and H. Zhou, "Fuzzy clustering for colour reduction in images," *Telecommun. Syst.*, vol. 40, no. 1, pp. 17–25, Feb. 2009.

[43] N. Papamarkos, A. E. Atsalakis, and C. P. Strouthopoulos, "Adaptive color reduction," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 1, pp. 44–56, Feb. 2002.
[44] C.-H. Chang, P. Xu, R. Xiao, and T. Srikanthan, "New adaptive color quantization method based on self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 237–249, Jan. 2005.
[45] C.-H. Wang, C.-N. Lee, and C.-H. Hsieh, "Sample-size adaptive self-organization map for color images quantization," *Pattern Recognit. Lett.*, vol. 28, no. 13, pp. 1616–1629, Oct. 2007.
[46] M. G. Omran, A. P. Engelbrecht, and A. Salman, "A color image quantization algorithm based on particle swarm optimization," *Informatica*, vol. 29, no. 3, pp. 261–270, 2005.
[47] C. Ozturk, E. Hancer, and D. Karaboga, "Color image quantization: A short review and an application with artificial bee colony algorithm," *Informatica*, vol. 25, no. 3, pp. 485–503, Jan. 2014.
[48] X.-S. Yang and X. He, "Firefly algorithm: Recent advances and applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, pp. 36–50, 2013.
[49] L. Brun and A. Trémeau, "Color quantization," in *Digital Color Imaging Handbook*, G. Sharma, Ed. Boca Raton, FL, USA: CRC Press, 2003, pp. 589–638.
[50] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York, NY, USA: McGraw-Hill, 1968.
[51] H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, "AntTree: A new model for clustering with artificial ants," in *Proc. Congr. Evol. Comput.*, Canberra, ACT, Australia, vol. 4, Dec. 2003, pp. 2642–2647.
[52] H. Azzag, G. Venturini, A. Oliver, and C. Guinot, "A hierarchical ant based clustering algorithm and its use in three real-world applications," *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 906–922, Jun. 2007.
[53] W. Burger and M. L. Burge, "Color quantization," in *Principles of Digital Image Processing*, vol. 54. London, U.K.: Springer, 2009.
[54] Z. Xiang and G. Joy, "Color image quantization by agglomerative clustering," *IEEE Comput. Graph. Appl.*, vol. 14, no. 3, pp. 44–48, May 1994.
[55] A. Weber. (2018). *USC-SIPI Image Database*. Accessed: Jun. 20, 2019. [Online]. Available: http://sipi.usc.edu/database/database.php/
[56] M.-L. Pérez-Delgado. (2018). *Images for Color Quantization*. Accessed: Jun. 20, 2019. [Online]. Available: http://audax.zam.usal.es/web/mlperez/cq.html
[57] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COMM-28, no. 1, pp. 84–95, Jan. 1980.

MARÍA-LUISA PÉREZ-DELGADO received the Engineering degree in computer science from the University of Valladolid, Spain, and the Ph.D. degree from the University of Salamanca, Spain.

She is currently a Professor with the Computer Science and Automatics Department, University of Salamanca. Her research interests include artificial intelligence, optimization, graph theory, and data mining. She has published several research articles and books related to these areas.

JESÚS ÁNGEL ROMÁN GALLEGO received the Engineering degree in computer science from the Pontifical University of Salamanca, in 2006, and the Ph.D. degree from the University of Salamanca, Spain, in 2016.

He is currently a Professor with the Computer Science and Automatics Department, University of Salamanca. His research interests include intelligent systems, artificial intelligence, and computer science applied to education. He has published several research articles and chapters in books related to these areas.