Computer Vision and Image Understanding 265 (2026) 104659

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

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

Grey wolf optimization for color quantization

María-Luisa Pérez-Delgado [a,*], Jesús-Ángel Román-Gallego [a], M. Emre Celebi [b]

[a] Universidad de Salamanca, Escuela Politécnica Superior de Zamora, Av. Requejo, 33, Zamora, 49022, Spain
[b] University of Central Arkansas, 201 Donaghey Ave., Conway, AR, 72035, United States

ARTICLE INFO

Communicated by Juergen Gall

Keywords:
Color quantization
Swarm-based methods
Image processing

ABSTRACT

Color quantization is an image processing operation that attempts to reduce the number of distinct colors used to represent an image without significant loss of quality. This operation is useful as an initial step to perform further processing on the image. The interest in this operation has led to several solution methods being proposed over the years. Within these methods several swarm-based methods have been used in recent years to solve the problem. This article discusses the application of one of these methods, called grey wolf optimization, to perform color quantization. This algorithm has generated good results when applied to a variety of optimization problems. Among the features that differentiate this algorithm from other swarm algorithms are its ability to converge toward better solutions, its speed, and the existence of a single control parameter. The article describes in detail how the grey wolf optimization method should be adapted to perform color quantization, so that it generates a quantized palette that allows the quantized image to be represented. In this case, each individual in the group represents a quantized palette, which is improved with the information provided by the group. The detailed description of the algorithm is complemented by an extensive testing section that compares the results of the proposed method to those of 16 others techniques. The results, based on the comparison of MSE, MAE, PSNR, SSIM, and runtime, show that grey wolf optimization can generate good quality images, better than most of the compared methods.

1. Introduction

Let us consider an RGB image where the pixels are arranged in a rows and b columns. Therefore, the image includes $n = a \times b$ pixels. The pixel placed in row i and column j is represented by three integer values from interval $[0, 255]$, corresponding to the amount of red (R), green (G) and blue (B) of the pixel: $p_{ij} = (R_{ij}, G_{ij}, B_{ij})$.

When considering the RGB color space, the palette used to represent an image can include up to $256^3 = 16777216$ distinct colors. The objective of color quantization methods is to define a new palette that includes fewer colors to represent the image, so that the resulting image is as similar to the original as possible. In this case, the new palette is called the quantized palette and the image represented with that palette is called the quantized image. Let us represent the quantized palette by a set of RGB colors: $P = \{c_1, \dots, c_q\}$, where q is the size of the palette (the number of distinct colors it contains).

Color quantization is a fundamental process in the field of digital image processing and analysis. By limiting the number of colors, a simpler and more manageable representation is achieved, which is essential for applications in computer vision, printing, graphic editing and data transmission, among others. It is a crucial technique that goes beyond simply reducing the color palette of an image, facilitating tasks such as compression (Chen et al., 2002; Chou and Liu, 2004), segmentation (Mignotte, 2008; An and Pun, 2014), image watermarking (Tsai et al., 2004; Kuo and Cheng, 2007), texture analysis (Losson and Macaire, 2015; Ponti et al., 2016), content-based image retrieval (Girgis and Reda, 2014; Liu et al., 2015), image dehazing (Berman et al., 2016), image matting (Chuang et al., 2001), color-to-greyscale conversion (Kuhn et al., 2008), saliency detection (Cheng et al., 2014), and skin detection (Phung et al., 2005).

Proper image quantization maintains visual quality and preserves essential details, helping to optimize resources and improve the automatic interpretation of images in various systems. The importance of color quantization lies in its ability to optimize visual representation across countless digital applications. From image compression to reduce file sizes and speed up web loading, to adapting graphics for devices with limited screen capabilities, color quantization is fundamental for ensuring an efficient and high-quality visual experience. Furthermore, it plays a vital role in graphic design, artistic visual effects, printing, and emerging fields like computer vision, where simplifying the color space can improve the performance of recognition algorithms.

* Corresponding author.
E-mail address: mlperez@usal.es (M.-L. Pérez-Delgado).

https://doi.org/10.1016/j.cviu.2026.104659
Received 27 May 2024; Received in revised form 31 October 2025; Accepted 13 January 2026
Available online 4 February 2026

Color quantization is a difficult problem for which different solution methods have been proposed over the years (Celebi, 2023). Traditionally, it has been tackled with methods ranging from histogram-based techniques to clustering algorithms like k-means. However, the nonlinear and often multimodal nature of the color space, coupled with the computational complexity that arises when dealing with large color palettes, presents significant challenges for these approaches. This is where swarm-based algorithms emerge as a powerful and promising alternative.

Swarm-based methods are optimization techniques inspired by the behavior observed in groups of natural individuals that exhibit intelligence when solving problems. These algorithms consider a population of primitive individuals that collaborate to solve a problem. Each individual independently can only perform very simple operations. However, when all individuals collaborate, they can solve complex problems. Swarm-based algorithms have shown their potential to solve many optimization problems (Hassanien and Emary, 2018). Their ability to efficiently explore complex search spaces and find solutions close to the global optimum, without relying on gradients or assumptions about problem convexity, makes them particularly well-suited for color quantization. By treating each color palette as an agent within a multidimensional search space, these algorithms can iteratively optimize the selection of representative color palettes, minimizing distortion or quantization error in the final image.

Color quantization can be formulated as an optimization problem, whose objective is to find the color palette that minimizes an error function. The objective function associated with the problem determines the quality of said palette. To do this, an index is calculated that defines the quality of the quantized image obtained by applying the quantized palette to represent the original image. Different image quality indices can be used for this purpose (Pérez-Delgado and Celebi, 2024). These indices can be classified as full-reference indices (which compare the original image with the quantized image to perform the calculation) and no-reference indices (which use only the quantized image).

The particle swarm optimization algorithm, proposed by Kennedy and Eberhart in 1995, is probably the most popular swarm-based method (Kennedy and Eberhart, 1995). Since then, many other methods have been proposed that mimic the behavior of different individuals, such as birds, fish, fireflies, dolphins, bacteria, frogs, bees, or bats (Chakraborty and Kar, 2017; Tang et al., 2021). Among the most recent methods is the grey wolf optimization (GWO) algorithm.

The GWO method was proposed by Mirjalili et al. in 2014, (Mirjalili et al., 2014). It is an optimization method inspired by the behavior of a group of grey wolves and tries to mimic the leadership hierarchy and hunting method of these wolves. There is a clear hierarchy and the leaders condition the behavior of the group. The algorithm considers a group of wolves that represent solutions to an optimization problem. The quality or fitness of the solution associated with a wolf is calculated from the objective function of the problem to be optimized. The method applies an iterative process that updates the solution represented by each wolf, taking into account the solution associated with the three best wolves in the population. GWO has several advantages over other swarm algorithms: it is easy to implement and has a simple search mechanism, only a parameter, high solution accuracy, and fast convergence speed.

GWO has been extensively studied in recent years (Hatta et al., 2019). It has attracted a lot of attention due to its simplicity and ease of implementation. Because of this, it has been applied to solve many problems, including feature selection (Emary et al., 2015, 2016; Hu et al., 2020; Medjahed et al., 2016; Vosooghifard and Ebrahimpour, 2015), neural network training (Amirsadri et al., 2018; Mirjalili, 2015; Mosavi et al., 2016; Muangkote et al., 2014), clustering (Aljarah et al., 2020; Tripathi et al., 2018; Zhang and Zhou, 2015), parameter estimation (Ali et al., 2017; Miao et al., 2020; Song et al., 2015) or forecasting (Altan et al., 2021; Tikhamarine et al., 2020). It has also been applied to several real engineering problems, including scheduling (Komaki and Kayvanfar, 2015; Lu et al., 2017), path planning (Dewangan et al., 2019; Zhang et al., 2016), dispatch problems (Kamboj et al., 2016; Sulaiman et al., 2015), controllers tunning (Precup et al., 2016; Sun et al., 2019), power systems optimization (Guha et al., 2016; Shakarami and Davoudkhani, 2016) and medical diagnosis (Li et al., 2017a; Sharma et al., 2019; Zhao et al., 2019; Pérez-Delgado and Román-Gallego, 2023). In addition, GWO has been used in the field of image processing to solve problems such as image classification (Ahmed et al., 2018), image segmentation (Khairuzzaman and Chaudhury, 2017; Li et al., 2017b) or steganalysis (Pathak et al., 2019; Shankar et al., 2019).

The research described in Schaefer et al. (2017) introduces the possibility of applying GWO to reduce the colors of an image. It uses the mean squared error as the objective function of the problem to be optimized and includes a penalty term to avoid creating unused colors in the quantized palette. This article includes a limited description of the specific operations considered to apply GWO to perform color quantization. On the other hand, it only includes computational results for 6 images reduced to 16 colors.

In addition to the article mentioned above, there are others that describe swarm-based methods applied to solve the color quantization problem. The methods applied in these articles mimic the behavior of particles, ants, bees, fireflies or frogs (Pérez-Delgado, 2020c). Published results indicate that these methods succeed in generating good quality quantized images.

Taking into account the above, it can be concluded that the GWO method can be applied to the color quantization problem and is likely to produce better results than other swarm models, as has been shown when applied to other problems. Therefore, the objective of this article is to present a detailed description of how the GWO algorithm can be applied to perform color quantization. The original GWO algorithm must be adapted to solve this specific problem. The article details these modifications and shows the influence of each modification in the final result. To evaluate the quality of the method, it is applied to a set of 24 color images commonly used in the color quantization literature. In addition, it is compared to 14 other color quantization techniques and 2 general optimization methods.

The rest of the article is organized as follows. First, popular methods applied to color quantization are described, including various swarm-based methods that have been applied to this problem. After this, the general GWO algorithm is analyzed and then its adaptation to perform color quantization is discussed. The article continues by presenting a set of experimental results obtained by the described method. The information is completed with a comparative analysis of the results obtained by other 16 methods. Finally, the conclusions of the article are presented.

The main contributions of the article are the following:

- The GWO method has been adapted to reduce the number of distinct colors in an image.
- Several modifications are proposed to avoid the two main drawbacks of the basic GWO algorithm: premature convergence and weak global searchability.
- Experimental results are given showing the different options that were considered to define the characteristics of the proposed solution.
- The results of the proposed method are compared to those of 14 other color quantization methods and 2 optimization methods. The set of compared methods includes both classical color quantization methods and various swarm-based methods. It has been shown that the proposed method generates better-quality images than most of its rivals.

2. Related work

Color quantization methods can be broadly classified into splitting methods and clustering-based methods. In general, splitting methods can generate a solution quickly, but clustering-based methods can obtain better solutions.

To define a quantized palette with q colors, splitting methods divide the color space into q regions and include a color in the palette to represent the pixels in each region. To accomplish this, these methods apply an iterative process in which a region is selected and then split to generate new regions.

The median-cut (MC) method (Heckbert, 1982) splits the region with the most pixels along the longest axis at the median point. The centroids of the final q regions define the quantized palette.

The variance-based (VB) method (Wan et al., 1990) selects the region with the largest weighted variance. The point that minimizes the marginal squared error is the splitting point, and the axis with the least weighted sum of projected variances is the splitting axis.

Wu proposed two splitting methods. The greedy orthogonal bi-partitioning method is based on the same idea as VB, but the splitting axis is the one that minimizes the sum of the variances of both sides (Wu, 1991). The second method sorts the colors of the image along their principal axis and then applies dynamic programming to divide the color space (Wu, 1992).

The octree (OC) method (Gervautz and Purgathofer, 1990) builds a tree with the pixels in the image and then applies a merging process to the leaves that represent fewer pixels until the tree includes only q leaves.

The binary splitting (BS) method (Orchard and Bouman, 1991) divides the region with the largest dominant eigenvalue along the principal axis of this region. The splitting point used in this case is the projection of the centroid to the selected axis.

The variance-cut (VC) method (Celebi et al., 2015) uses the binary splitting strategy to split the region with the largest sum of squared error along the coordinate axis with the largest variance at the mean point.

On the other hand, clustering-based methods create q subsets or groups of pixels of the original image that have similar color. Several general clustering methods have been adapted to perform color quantization. These methods include the k-means algorithm, neural networks and swarm-based methods.

K-means is probably the most popular clustering method. There are several articles where it has been applied to perform color quantization (Abernathy and Celebi, 2022; Bounds et al., 2024; Celebi, 2009, 2011; Hu and Su, 2008; Kasuga et al., 2000; Thompson et al., 2020). This method selects q initial centroids that define the quantized palette and applies an iterative process to improve them (Celebi et al., 2013). At each iteration, each pixel is associated with the closest centroid, and each centroid is then recomputed as the average value of all pixels associated with it in the previous operation.

The adaptive distributing units (ADU) method (Celebi et al., 2014) is based on the competitive learning paradigm. The iterative process starts with a palette that includes a single cluster whose centroid is the average of the pixels in the image. At each iteration, a pixel of the input image is associated with the closest color in the current quantized palette. When the number of pixels associated with a color in the palette reaches a threshold, a new element is added to the palette with the same initial color. The process concludes when the palette includes q colors.

The neuquant (NQ) method defines a quantized palette using a neural network (Dekker, 1994). The final weights of a one-dimensional self-organizing feature map including q neurons define such palette.

In addition to the solution based on the GWO method described in the introduction, various swarm-based algorithms have been applied to solve the color quantization problem, including artificial ant algorithms, the shuffled-frog leaping algorithm (SFLA), the firefly algorithm (FA), the particle swarm optimization (PSO) method, and the artificial bee colony (ABC) algorithm.

The method described in Pérez-Delgado (2015), called ant-tree for color quantization (ATCQ), defines a tree of ants. Each ant represents a pixel of the original image that is connected to the tree taking into account the similarity between the color of the ant and the color of the tree node to which they are trying to connect. Once all the ants have been connected, each node of the second level defines a color of the quantized palette and is the root node of a subtree of ants (pixels) that will be represented in the quantized image by the color of that root node. The algorithm includes a parameter, α, that allows calculating a threshold that is used to decide if the similarity between an ant and a node is enough to connect the ant to the node.

ITATCQ is the iterative version of ATCQ (Pérez-Delgado, 2018b). ATCQ creates the initial tree. Then, each iteration starts by disconnecting all the ants from the tree and then reconnecting them. The ITATCQ method was revised in Pérez-Delgado (2021), including several modifications to the original method. In this case, the ants are processed randomly (rather than sequentially), and the α parameter is reduced over the course of iterations. The results showed that the quantized image is better and that the value of the α parameter has less influence on the final results.

The ATCQ+FA method combines FA with ATCQ to perform color quantization (Pérez-Delgado, 2018a). Each firefly represents a quantized palette and ATCQ is applied both to calculate the fitness of the solution it represents and to update it.

The PSO method was combined with k-means in Omran et al. (2005), applying k-means to improve several randomly selected particles before calculating their fitness. PSO was also used in Pérez-Delgado (2020a), although in this case ATCQ was used to compute the fitness of each particle and at the same time improve the palette represented by the particle. Tests conducted with both methods showed that the second method obtains better results in less time.

Two color quantization methods based on the use of the ABC algorithm were proposed in Ozturk et al. (2014), Pérez-Delgado (2019b). Similar to what was described for the two PSO-based methods, these articles combine ABC with k-means (Ozturk et al., 2014) and ATCQ (Pérez-Delgado, 2019b), which are used to compute the fitness of all the food sources except the candidate sources. The computational results showed that the execution time was considerably reduced by using ATCQ.

SFLA was used for color quantization in Pérez-Delgado (2019a). The algorithm works on a subset of pixels of the original image in order to reduce execution time.

Swarm-based methods have also been applied to the solution obtained by other color quantization methods. This defines a two-stage method that first applies a splitting method to obtain an initial quantized palette and then applies a clustering-based method that improves said palette. This approach was used to combine the BS method with ATCQ in Pérez-Delgado and Román-Gallego (2020) and then with ITATCQ in Pérez-Delgado (2020b). On the other hand, ATCQ was combined with the greedy orthogonal bi-partitioning method in Pérez-Delgado and Román-Gallego (2019). The advantage of these solutions is that they generate good quality images fast.

Other swarm-based solutions to the color quantization problem are described in Pérez-Delgado (2020c).

Several works can be found in the literature that apply other metaheuristics to solve this problem. Among the techniques applied are simulated annealing (Nolle and Schaefer, 2007; Schaefer and Nolle, 2015), differential evolution (Schaefer and Nolle, 2006; Su and Hu, 2013; Hu et al., 2016), variable neighborhood search (Hansen et al., 2007), genetic algorithms (Freisleben and Schrader, 1997; Roberto e Souza et al., 2020; Scheunders, 1997; Taşdizen et al., 1998) and evolution strategies (Carro-Calvo et al., 2010; González et al., 2000).

It is not the intention of this section to enumerate all existing color quantization methods. For a comprehensive review of methods developed over the past 40 years, the reader is encouraged to consult (Celebi, 2023).

3. GWO algorithm

Let us consider an optimization problem defined in a solution space of dimension r, where $f()$ denotes the objective function of the problem. The solution to the problem, $S = (s_1, \ldots, s_r)$, is the value that minimizes (or maximizes) the objective function.

To solve the optimization problem by GWO, a population of wolves is considered. Let N denote the size of the population. Each wolf i has a position associated with it, $X_i = (X_{i1}, X_{i2}, \ldots, X_{ir})$, that represents a solution to the problem. Said position is modified over the course of iterations of the GWO algorithm; therefore $X_i(t)$ denotes the position of the wolf i at iteration t. The quality or fitness of said solution is calculated by applying the objective function of the problem to the position associated with the wolf, $f(X_i)$. The three wolves with the best fitness are used to guide the movement of the swarm. The three best wolves are named wolf α, β and δ, respectively, and their positions are $X_\alpha(t)$, $X_\beta(t)$ and $X_\delta(t)$. The α wolf is the one with the best position, that is, the wolf whose position generates the best value for the objective function of the problem.

Algorithm 1 Grey Wolf Optimization

```
Define initial population
Compute the fitness of each wolf
Identify the $\alpha$, $\beta$ and $\delta$ wolves
Store $X_\alpha(0)$ as the global best solution (initial)
for $t = 1$ to $TMAX$ do
    Update the positions of the wolves by Eq. (1)
    Compute the fitness of each wolf
    Identify the $\alpha$, $\beta$ and $\delta$ wolves
    Update $a$ by Eq. (10)
    Update the global best solution
end for
```

The main operations of this method are listed in Algorithm 1. This algorithm includes only one parameter, a, which is initialized before the operations start and is updated in each iteration. The variable $TMAX$ represents the maximum number of iterations of the GWO algorithm.

The first operation of the algorithm selects initial values for the wolves in the population, $\{X_1(0), X_2(0), \ldots, X_N(0)\}$. After this, the fitness of the wolves is computed by applying the objective function of the problem. Then, the three wolves with the best fitness are selected as the α, β and δ wolves, respectively. In addition, the current solution associated with the α wolf is stored as the best initial solution to the problem.

After the initialization stage, the algorithm applies an iterative process to improve the solution obtained by the swarm. At each iteration t, the position of each wolf is updated taking into account the position of the three best wolves. The new position of a wolf i, $X_i(t)$, is computed by Eq. (1), where the vectors $X1$, $X2$ and $X3$ are computed by Eqs. (2)–(4), respectively; ($\otimes$ represents the Hadamard product).

$$X_i(t) = (X1 + X2 + X3)/3 \tag{1}$$

$$X1 = X_\alpha(t-1) - A_1 \otimes D_\alpha \tag{2}$$

$$X2 = X_\beta(t-1) - A_2 \otimes D_\beta \tag{3}$$

$$X3 = X_\delta(t-1) - A_3 \otimes D_\delta \tag{4}$$

The equations applied to compute $X1$, $X2$ and $X3$ consider the current position of one of the three leading wolves ($X_\alpha(t-1)$, $X_\beta(t-1)$ or $X_\delta(t-1)$), along with two additional elements. A_1, A_2 and A_3 are vectors of size r computed by applying Eq. (5) three times, where r_1 is a vector of size r whose components take random values from the interval $[0,1]$ and a is a user-defined parameter.

$$A = 2\ a\ r_1 - a \tag{5}$$

On the other hand, D_α, D_β and D_δ are computed by Eqs. (6)–(8), respectively, where C_1, C_2 and C_3 are computed by applying Eq. (9) three times and r_2 is a vector of size r whose components take random values from the interval $[0,1]$. The operator $|\cdot|$ used in these equations indicates that the absolute value of each component of the resulting vector is taken.

$$D_\alpha = |C_1 \otimes X_\alpha(t-1) - X_i(t-1)| \tag{6}$$

$$D_\beta = |C_2 \otimes X_\beta(t-1) - X_i(t-1)| \tag{7}$$

$$D_\delta = |C_3 \otimes X_\delta(t-1) - X_i(t-1)| \tag{8}$$

$$C = 2r_2 \tag{9}$$

Once the positions of the wolves have been updated, the fitness of the new positions is computed and this allows the identification of the new three best wolves in the population. If the position of the α wolf in the current iteration improves the global best solution stored so far, that information is replaced by the position of the current α wolf.

Before completing an iteration, the algorithm updates the value of the a parameter. This parameter is reduced over the course of iterations. The original article of Mirjalili proposes setting the initial value equal to 2 before applying the algorithm and then using Eq. (10) to update a, so as the parameter decreases linearly from 2 to 0 over the course of iterations.

$$a = 2\left(1 - \frac{t}{TMAX}\right) \tag{10}$$

When the algorithm terminates, the solution to the problem is given by the global best solution.

Several variants of the original algorithm have been proposed to try to improve the performance of the method. Some variants modify the GWO update mechanism to improve the balance between exploration and exploitation. For this purpose, several authors update the parameters dynamically (Long et al., 2017; Mittal et al., 2016), while others define new methods to update the individuals (Dudani and Chudasama, 2016; Gupta and Deep, 2019; Malik et al., 2015; Rodríguez et al., 2017).

4. GWO applied to color quantization

This section describes how the GWO algorithm has been adapted to perform color quantization. The resulting algorithm is called GWO+ATCQ, to differentiate it from the original GWO.

To solve the color quantization problem, the search agents in the group represent quantized palettes. Let consider a group of N search agents. The quantized palette corresponding to the agent i includes q RGB colors: $X_i = \{(R_{i1}, G_{i1}, B_{i1}), \ldots, (R_{iq}, G_{iq}, B_{iq})\}$. Therefore, the algorithm handles N quantized palettes: $\{X_1, \ldots, X_N\}$. In an 8-bit color system, the red, green, and blue components of each color in a palette are represented by integer values between 0 and 255.

In order to describe the operations, this section includes several figures showing the evolution of a swarm of six search agents trying to reduce an image to 64 colors. The results correspond to the first image of the test set used in Section 5 of this article. The figures show the fitness of each agent over 20 iterations of the algorithm. The results corresponding to the iteration labeled 0 represent the fitness of the initial palettes. To perform a fair comparison, the same initial palettes were used for all the examples.

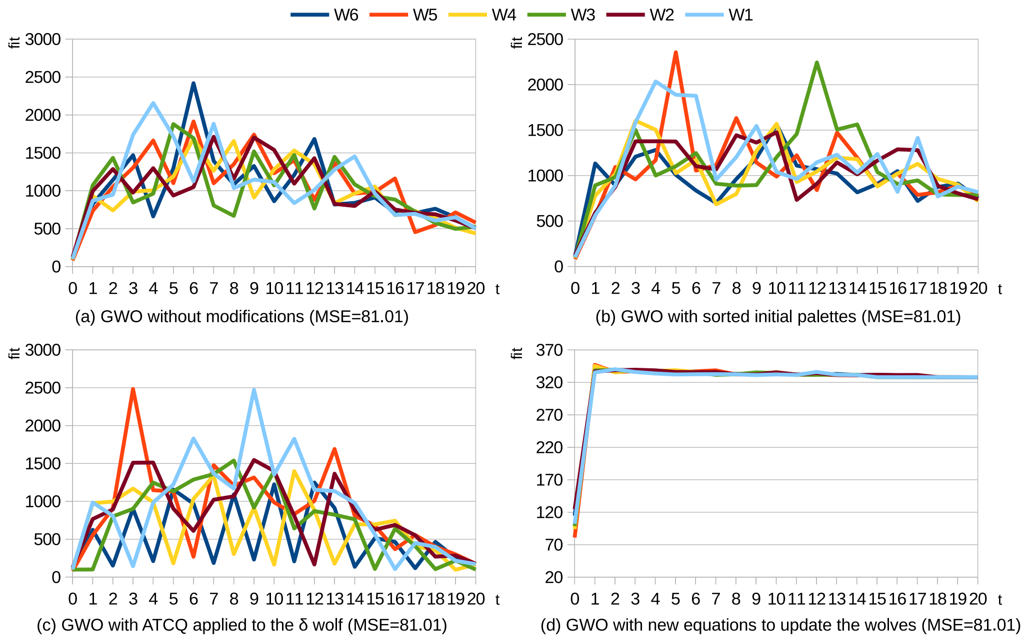

Fig. 1. Evolution of a swarm of 6 agents over 20 iterations of the GWO algorithm – original GWO and several modifications (images reduced to 64 colors).

Algorithm 2 GWO+ATCQ

```
Sample the original image
Define initial population
Compute the fitness of the search agents by Eq. (11)
Identify the best three search agents
Store the initial best solution (Set $P_{gp} = X_\alpha(0)$)
for $t = 1$ to $TMAX$ do
    Update the position of the search agents
    Adjust the palette components to $[0, 255]$
    Improve the palette of the $\delta$ search agent
    Compute the fitness of the search agents by Eq. (11)
    Identify the best three search agents
    Update $a$ by Eq. (10)
    Update the global best solution $P_{gp}$
end for
Adjust the colors of $P_{gp}$ by using the entire original image
```

4.1. The steps of the solution method

Algorithm 2 shows the steps of GWO+ATCQ. The fitness of a palette i is denoted fit_i and it is computed by applying Eq. (11), which is described in the next section.

First of all, a sampling process is applied to the original image in order to reduce the size of the set of pixels used to apply the algorithm. Next, the initial N palettes are defined and their elements are sorted by ascending value of the color components. Once the fitness of the N search agents has been computed, the three best agents are determined. The palette corresponding to the α agent is stored as the initial best solution to the problem. Let P_{gp} denote the best quantized palette found by the group of agents.

After completing the initialization stage, the algorithm performs an iterative process to improve the palettes represented by the search agents. The steps applied during each iteration are essentially those described for the GWO algorithm. Nevertheless, the operations applied to update the positions (palettes) of the agents are modified, as will be described in a later section. In summary, modified equations are included to compute the new position of an agent and the ATCQ method is also applied to improve the solution represented by the δ agent.

The a parameter was assigned the same values proposed in the original GWO method. Therefore, the initial value was set to 2 and it was updated by Eq. (10). The final operation of each iteration determines whether the group has found a better solution than the one considered the best up to the current iteration (P_{gp}). If the fitness of the α agent identified at the current iteration is better than the fitness of P_{gp}, the palette corresponding to agent α is saved as the global best solution.

It should be noted that when the positions of the search agents are updated, the resulting values for each component must fit to the integer interval $[0, 255]$ that defines the valid values for the components of an RGB color.

To sum up, the main modifications and features considered to perform color quantization using GWO are the following:

1. the fitness function.
2. sampling of the original image.
3. initialization of the population.
4. sorting of the initial solutions represented by the search agents.
5. updating the positions of the agents.
6. improvement of the δ agent by applying ATCQ.

These modifications are discussed in the following sections.

Before starting with this description, Fig. 1 shows the results for the original GWO and also for this method with several of the modifications proposed in the article, but applied independently of each other. The results obtained by the basic GWO appear in Fig. 1(a) (case in which no modifications of the proposed method are applied). It can be observed that the solutions of the search agents oscillate a lot. The results shown in Fig. 1(b) correspond to the case in which the initial palettes were sorted. Then, Fig. 1(c) shows the results obtained when ATCQ is applied to the δ agent. Finally, Fig. 1(d) shows the results obtained for the case that considers new equations to update the position of the agents. The MSE (mean squared error) value shown next to each figure represents the fitness of the best palette found by the group of agents at the end of the iterations.

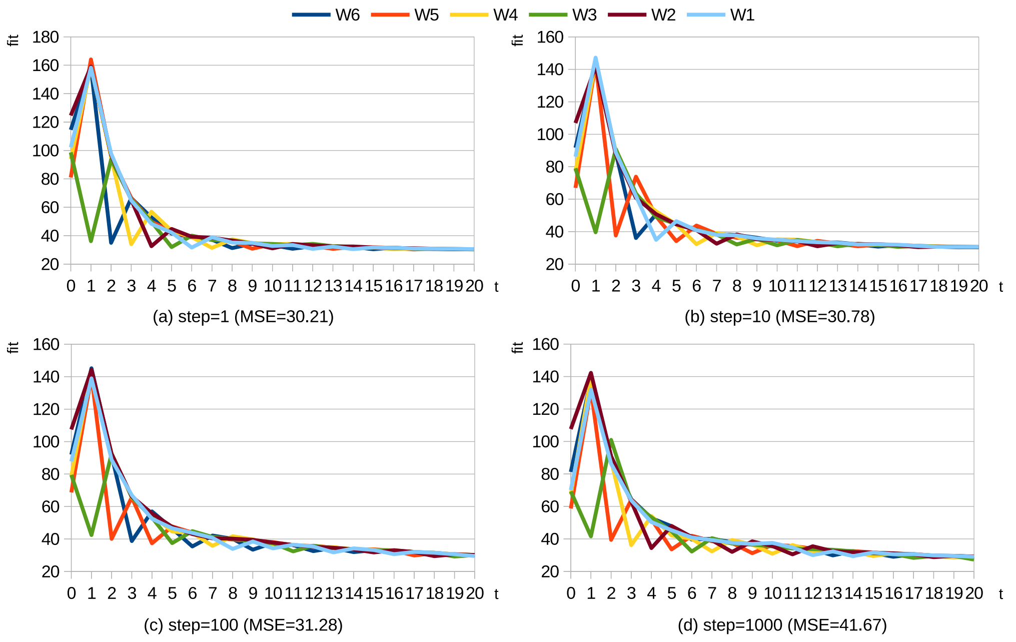

Fig. 2. Evolution of a swarm of 6 agents over 20 iterations of GWO+ATCQ with sampled image (images reduced to 64 colors).

4.2. The fitness function

The function used to quantify the fitness of an agent is the mean squared error, computed by Eq. (11). In this equation p_{ij} represents the color of a pixel of the original image placed in row i and column j, while p'_{ij} is the pixel of the quantized image which occupies the same position as p_{ij}. The smaller the MSE value, the better the solution represented by the agent.

$$MSE = \frac{1}{ab}\sum_{i=1}^{a}\sum_{j=1}^{b}\|p_{ij} - p'_{ij}\|^2 \tag{11}$$

Before calculating the MSE, it is required to select a color from the quantized palette to represent each pixel of the image. This operation is performed by selecting for each pixel of the original image the closest color from the quantized palette.

4.3. Sampling

In general, swarm-based methods require computing the fitness of all the individuals in the population at each iteration of the algorithm. When applied to the color quantization problem, it is necessary to perform a calculation involving the pixels of the original image. For example, the objective function proposed in this article requires this operation. Such an operation is time consuming for large images.

When the SFLA method was applied to color quantization, it was proposed to use a sampled image to reduce the execution time of the algorithm (Pérez-Delgado, 2019a). The computational results included in that article showed that the method can obtain good results when the original image is sampled. This allows the method to be fast, while producing good quality results. For this reason, the method proposed in this article uses the same sampling strategy described for the application of frogs to the same problem.

The pixels of the original image are sampled at a distance defined by the *step* parameter. Said pixels are processed sequentially from left to right and from top to bottom, so the following sequence of pixels is processed: $\{p_{11}, p_{12}, \ldots, p_{1b}, p_{21}, \ldots, p_{2b}, \ldots, p_{a1}, \ldots, p_{ab}\}$. That is, the image pixels are processed considering that all of them are included in a linear list. The first pixel selected is p_{11}, then, the following pixels are selected considering a distance equal to *step* in the processing sequence. That is, the second pixel selected is the one at position $step + 1$ in the linear list of pixels (assuming that $step < ab$).

The sampled image includes a subset of n' pixels of the original image, with $n' < n$. When the algorithm requires to compute the fitness of a search agent, this computation will be based on the sampled image, thus speeding up the process. Obviously, when $step = 1$, all the pixels of the original image are taken without any sampling.

Fig. 2 shows the results for four sampling steps. It can be observed that the resulting MSE is very similar for $step = 1$ and $step = 10$. As will be described in the computational results section, the execution time decreases as *step* increases, so these two *step* values will be used to draw the figures that describe the remaining operations in this section. Specifically, the average execution time obtained for 20 independent tests in this case was 12035 millisecond for $step = 1$, 1420 ms for $step = 10$, 322 ms for $step = 100$ and 217 ms for $step = 1000$. It can be observed that the fitness value drawn in the figure is larger when the image was not sampled ($step = 1$). It should be noted that the values drawn in this case correspond to the entire image, but the values for the other cases correspond to the subset of sampled pixels. Therefore, the smaller the sampled subset, the smaller the error of the quantized palette computed at each iteration of the algorithm. Nevertheless, the MSE value indicated for each subfigure is computed for the entire image with the best palette found along the iterations. Therefore, although when comparing the 4 subfigures it may seem that the case corresponding to the unsampled image generates larger errors, the MSE values clearly show that the error of the final quantized image increases when a sampled image is used.

4.4. Initialization of the group of agents

The algorithm requires defining N initial palettes to associate them to the agents in the group. Each palette is defined by selecting q random pixels from the original image and taking their color as a member of the quantized palette. To perform this operation, the complete original image (without sampling) is considered, to give more diversity to the initial values.

After this, the q colors that define each palette are sorted by ascending value of the three color components. To perform this operation, the colors are first sorted by their red component. Then, the colors with the same red value are further sorted by the green component. Finally,

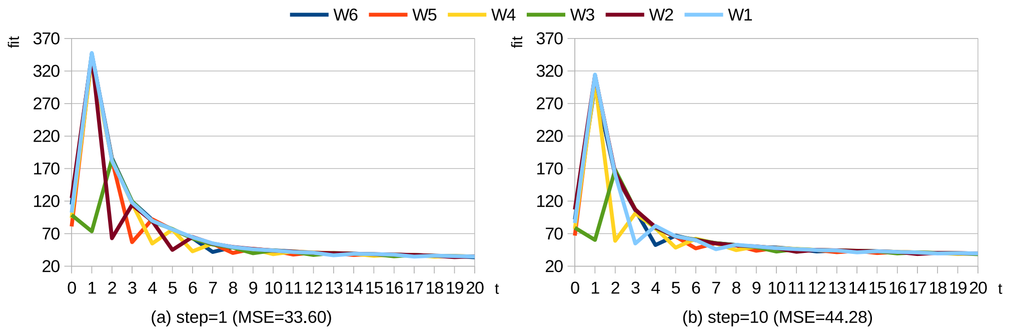

Fig. 3. Evolution of a swarm of 6 agents over 20 iterations of GWO+ATCQ when the initial palettes are not sorted (images reduced to 64 colors).

11

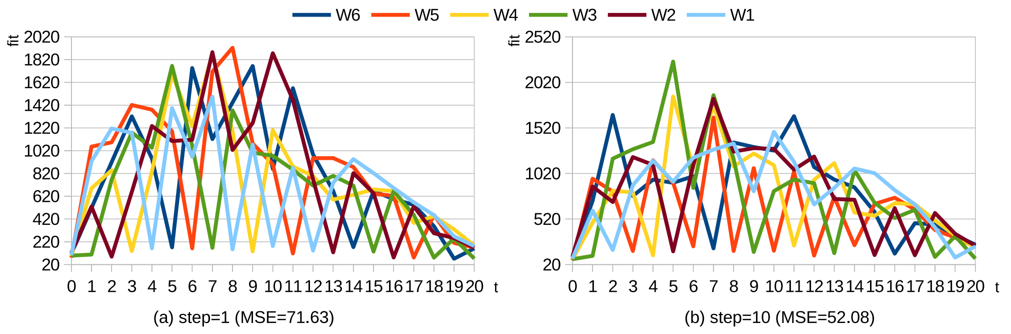

Fig. 4. Evolution of a swarm of 6 agents over 20 iterations of GWO+ATCQ when Eqs. (2) to (4) are used to compute $X1$, $X2$ and $X3$ (images reduced to 64 colors).

those colors with the same red and green value are sorted by the blue component. In this way, the update of the agents in the initial stage considers similar colors to perform the operation defined in GWO.

Fig. 3 shows the results for the proposed method when the initial palettes are not sorted. When compared to the results of Figs. 2(a) and 2(b), where sorted palettes were used for the same *step* values, it can be observed that the final images have better quality in both cases.

4.5. Updating the positions of the agents

To update the palette associated with an agent, the original equations used by GWO have been adapted. Computational experiments showed that these equations cause the palette values to fall outside the valid range $[0, 255]$ many times. This limits the effect of the equations, since the computed values must always be adjusted to that interval to represent valid RGB components. For this reason, the quality of the palette represented by an agent is used to update its position.

When the position of an agent i must be recomputed using Eq. (1) of GWO, Eqs. (2) to (4) are replaced with Eqs. (12) to (14), respectively, where fit_i represents the fitness of the current position of said agent (that is, the fitness of $X_i(t-1)$). When applying Eq. (1) to calculate the new palette associated with an agent, it must be observed that the resulting vector is a vector of integers. Therefore, the decimal parts of all the components of the vector X_i calculated using Eq. (1) are truncated.

$$X1 = X_\alpha(t-1) - (A_1 \otimes D_\alpha)/fit_i \tag{12}$$

$$X2 = X_\beta(t-1) - (A_2 \otimes D_\beta)/fit_i \tag{13}$$

$$X3 = X_\delta(t-1) - (A_3 \otimes D_\delta)/fit_i \tag{14}$$

In this way, the quality of the palette represented by the agent i influences the new position of said agent. The better the palette associated with agent i, the lower its MSE. If the agent represents a good palette, fit_i is small and the final term of the above equations is larger than when the agent represents a bad palette. Therefore, the contribution of agent i to its new position is greater when its current position is good than when it is bad. In the latter case, the position of the three leading agents is the one that most influences the new position of the agent i.

Fig. 4 shows the results obtained when the factor described in this section is not applied to GWO+ATCQ. It is clear that the solutions oscillate for many iterations but this does not help to generate a final result better than the one obtained during the initial iterations. Therefore, it is observed that this modification applied to the algorithm greatly influences the final result.

As stated above, once a color in a quantized palette is updated, it must always be adjusted so that all three of its components (red, green, blue) are integer values between 0 and 255.

4.6. Improvement of the δ agent

The GWO method includes two main stages. The first one is devoted to the exploration of new solutions, while the second one is devoted to the exploitation of the previous knowledge.

The variables A and C influence the exploration and exploitation capacity of the method. When $|A|$ >1, the exploration is intensified. In addition, when C >1, the exploration ability of the agents is also enhanced. On the contrary, when $|A|$ <1 and C <1, the exploitation ability is enhanced. The value of A decreases over the course of algorithm iterations, which enhances the exploitation ability. On the other hand, C is randomly generated at each iteration and this contributes to balance exploration and exploitation at all times.

To improve the results of GWO applied to color quantization, the exploitation phase is strengthened by improving the solution represented by the best agents. Since these agents condition the new position of the other individuals in the group, improving the leader agents will influence the entire group.

For this purpose, GWO+ATCQ adjusts the position of the δ agent by applying a reduced variant of the ATCQ algorithm introduced in Pérez-Delgado (2015). In this case the ATCQ variant that builds a 3-level tree is considered. Furthermore, the algorithm is applied to a tree that already includes the maximum number of allowed children of the root node, so the α parameter of the ATCQ algorithm is not required.

Algorithm 3 shows the ATCQ operations applied to improve the quantized palette represented by an agent. The algorithm considers the current palette of the agent as input information and updates that palette as a result of applying its operations.

First, the root node of the tree is created. Then, q children of the root are created, $\{S_1, \dots, S_q\}$, and each is assigned a color of the input palette. The two values used to define the color of a node are the sum of the colors of the ants included in the subtree, sum_k, and the number of ants in the subtree, $items_k$. When the child S_k is created, both variables are initialized, so as sum_k stores a color of the input palette and $items_k$ is set to 1.

Algorithm 3 ATCQ to improve a search agent

```
DATA: $P = \{c_1, ..., c_q\}$ (palette of the agent)
Create the root node of the tree
for $k = 1$ to $q$ do
    Create the child $S_k$ of the root
    Set $sum_k = c_k$
    Set $items_k = 1$
end for
Create the list of ants
for each and $h_{ij}$ do
    Select the child of the root node most similar to $h_{ij}$, $best$
    Connect $h_{ij}$ to the subtree $best$
    Set $sum_{best} = sum_{best} + h_{ij}$
    Set $items_{best} = items_{best} + 1$
end for
for $k = 1$ to $q$ do
    Set $c_k = sum_k \ / \ items_k$
end for
```

Each pixel p_{ij} of the sampled image is represented by the ant h_{ij} and all the ants are inserted into a list. Next, an iterative process is applied to connect all the ants to the tree. In this case the ants are randomly selected from the set of ants.

After selecting an ant h_{ij}, the most similar node of the second level of the tree must be identified. The color of a node S_k used to compare it to an ant is computed by $sum_k/items_k$. Eq. (15) is used to compute the similarity between ant h_{ij} and the color of the node S_k, where $dist$ represents the Euclidean distance. Let $best$ denote the child of the root node most similar to h_{ij}. Once this node is identified, h_{ij} is connected to the subtree with root S_{best} and the two variables associated with said node are updated: the color of h_{ij} is added to the sum of colors sum_{best} and the number of ants in the subtree is increased by one.

$$Sim(h_{ij}, S_k) = \frac{1}{1 + dist(h_{ij}, S_k)} \tag{15}$$

Once all the ants have been connected to the tree, the input palette values are updated by using the colors of the nodes in the second level of the tree.

As stated above, the operations of Algorithm 3 are applied to the δ agent. Nevertheless, other possibilities were also analyzed before selecting this one. Fig. 5 compares the results obtained when ATCQ is applied to several of the best agents.

The possibility of applying the improvement to the three best agents was analyzed (Figs. 5(a) and (b)). Nevertheless, the experimental results show that the exploration process of the GWO method loses its effect if ATCQ is applied to these three agents. On the other hand, the execution time increases when upgrading three agents instead of just one. The average time for 20 independent tests was 16368 ms for the case with $step = 1$ and 1792 ms for the case with $step = 10$. In contrast, when only the δ agent is improved, the values are reduced to 12035 and 1420 millisecond, respectively.

Next, the option of updating only one of the three agents was considered, thus allowing the exploration process of GWO and also reducing the total execution time of the method. Fig. 2 includes the results for the case in which only the δ agent is improved. In addition, Figs. 5(c) and (d) show the results for the case that only improves the α agent, while Figs. 5(e) and (f) show the case that only improves the β agent. When only the best agent is improved, it is clear that the group of agents does not perform exploration. The exploration increases when the β agent is improved instead of the α agent. In addition, it was observed that when the α agent is improved, the same agent is the best in all the iterations. However, when one of the other two agents is improved, the three best individuals selected at each iteration are different. This shows that when agents β or δ are improved the exploration in GWO is maintained and also that the improvement will affect different agents in different iterations. Finally, it was decided to apply ATCQ to the δ agent instead of the β agent because this helps to improve the worst of both solutions.

It was also analyzed when to start applying ATCQ (Fig. 6). GWO+ATCQ applies the modification from the initial iteration of the algorithm, but the figure shows the case in which only a subset of iterations are considered. The variable A causes exploration to be enhanced in the initial half of the GWO iterations and exploitation to be enhanced in the second half. For this reason, the result obtained when ATCQ is applied to the agent δ the second half of the iterations of GWO+ATCQ was analyzed. As can be observed (Figs. 6(a) and (b)), the results are worse. The evolution of the group of agents shows that in the first half of the iterations the results are worse than the initial ones. This evolution does not improve until the stage is reached where ATCQ begins to be applied. In addition, a large descent in the graph is observed when ATCQ begins to be applied. Therefore, the figures show that the algorithm wastes the initial set of iterations, since it fails to improve the initial result until ATCQ begins to be applied. On the other hand, the case in which ATCQ is not applied was also considered (Figs. 6(c) and (d)). In this case, it is clearly observed that the trend of the initial iterations is maintained until the end of the iterations. Certainly, the best palette found by the algorithm belongs to the initial set of palettes selected to apply the algorithm; that is, the iterations of the method do not contribute to improve the initial solutions.

4.7. *Final quantized palette*

When the original image is sampled, the algorithm defines a quantized palette for the pixels in the sample. If the sampled image includes only a few colors, the final palette might not adequately represent the pixels not included in the sample. For this reason, after defining the palette by applying the algorithm, said palette is adapted to the total set of pixels of the original image. With this purpose, the color of the palette that is most similar to each pixel of the original image is determined (the similarity is computed based on the Euclidean distance). After this, each color in the quantized palette is recomputed as the average value of all pixels in the original image associated with it in the previous operation. This operation improves the quality of the final image, especially when the sampled image includes a limited number of colors.

5. Results and discussion

5.1. *GWO+ATCQ results*

The proposed method was applied to 24 RGB color images available at Franzen (2019), each including 393216 pixels. Fig. 7 shows the images.

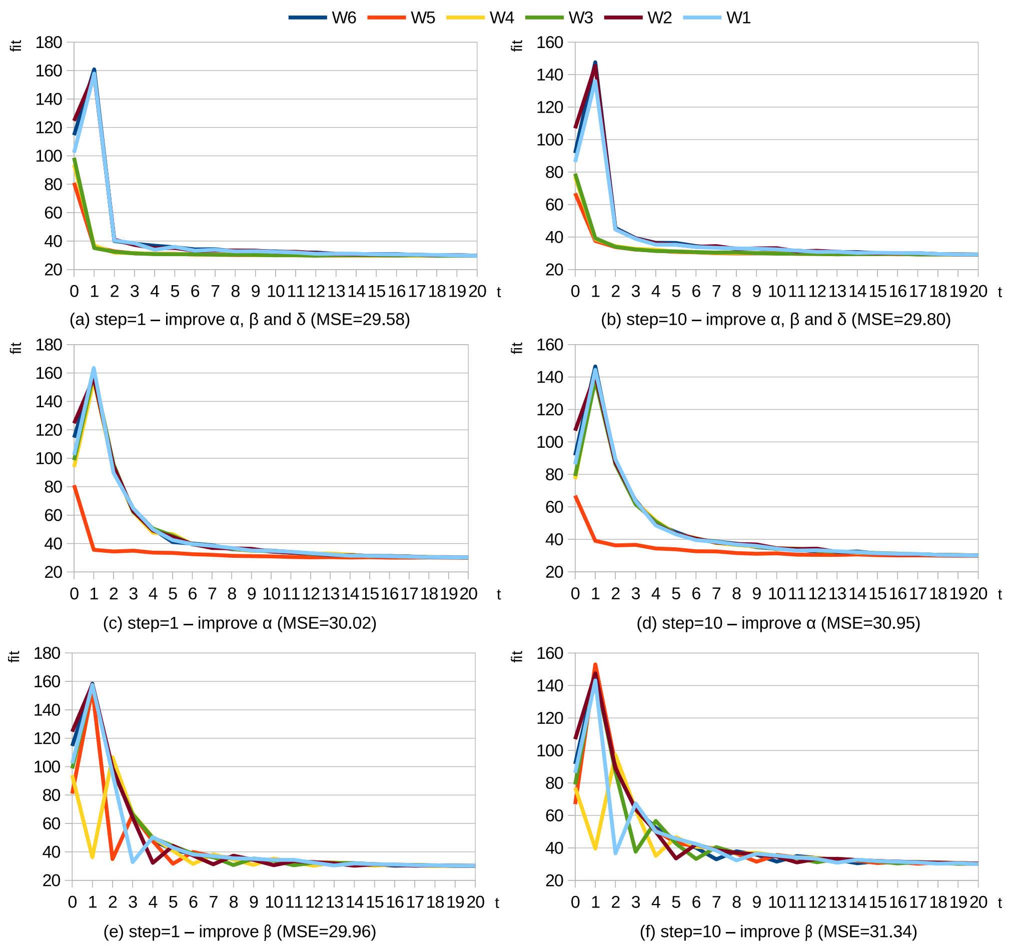

Fig. 5. Evolution of a swarm of 6 agents over 20 iterations of GWO+ATCQ when the palettes of several wolves are adjusted by ATCQ (images reduced to 64 colors).

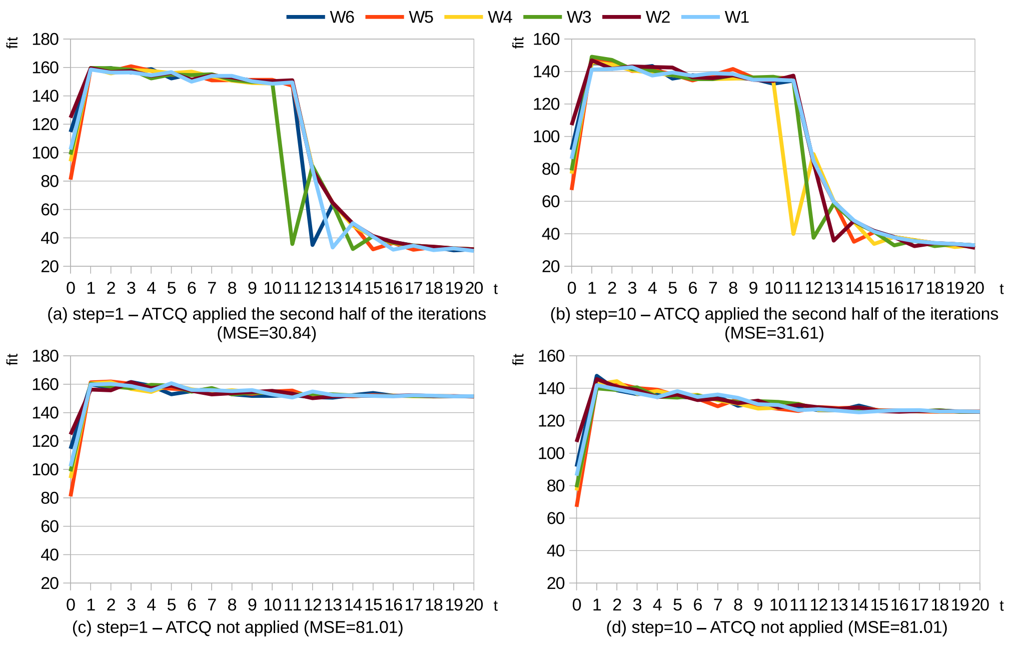

Fig. 6. Evolution of a swarm of 6 agents over 20 iterations of GWO+ATCQ when the palette of the δ wolf is improved by ATCQ for several iterations (images reduced to 64 colors).

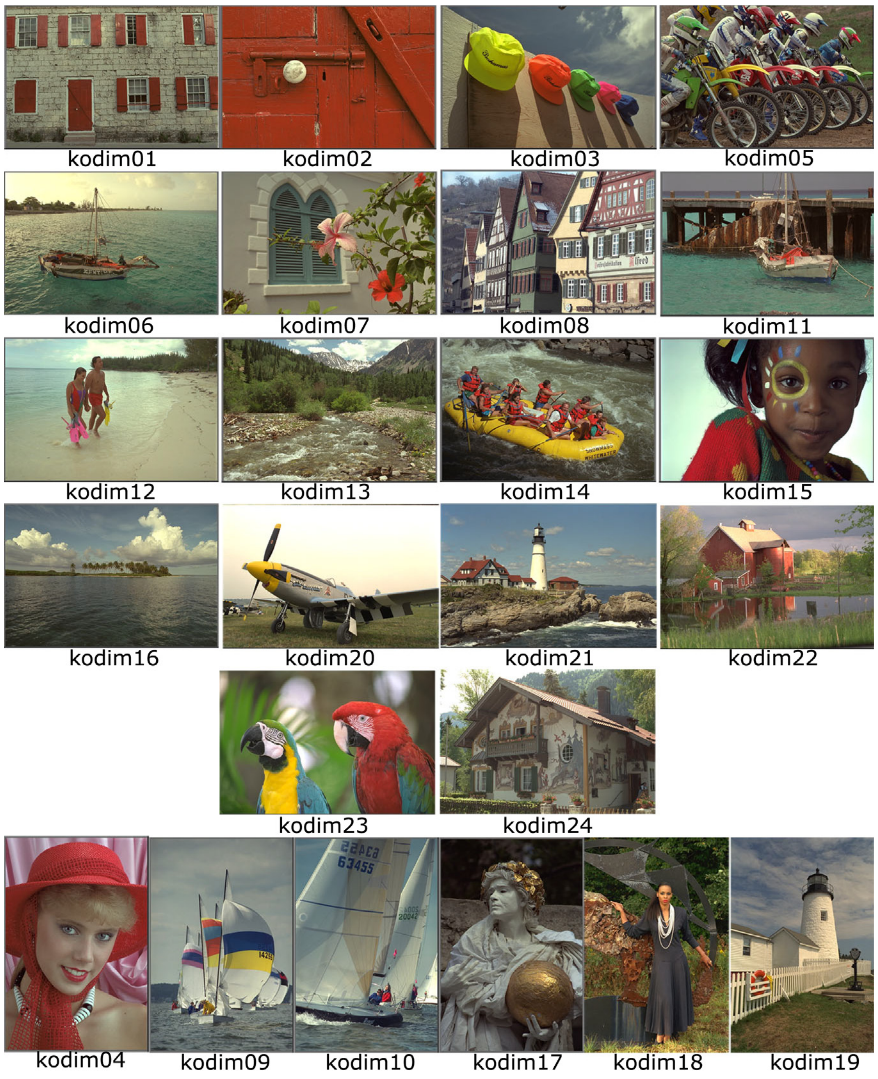

Fig. 7. Images in the test set.

Table 1
MSE, MAE, PSNR, SSIM and execution time (T) for the set of quantized images with 16, 64 and 256 colors obtained by GWO+ATCQ (*step*: value used to sample the original image; *av*: average; *dev*: standard deviation).

		16 colors		64 colors		256 colors	
	step	*av*	*dev*	*av*	*dev*	*av*	*dev*
MSE	10	206.944	84.343	61.408	27.237	23.519	10.458
	1	204.771	84.735	59.757	26.809	22.189	9.943
MAE	10	18.015	3.885	9.452	2.277	5.663	1.385
	1	18.006	3.879	9.386	2.281	5.568	1.368
PSNR	10	25.318	1.738	30.659	1.895	34.823	1.878
	1	25.376	1.772	30.790	1.929	35.081	1.886
SSIM	10	0.891	0.041	0.957	0.023	0.982	0.011
	1	0.891	0.041	0.958	0.023	0.983	0.011
T	10	413	21	1437	55	4983	162
	1	3481	187	11 997	459	43 647	1555

Four error measures were computed for the resulting quantized images: the MSE, the mean absolute error (MAE), the peak signal-to-noise ratio (PSNR), and the structural similarity index measure (SSIM). These error measures are commonly used in research articles related to color quantization methods (Pérez-Delgado and Celebi, 2024).

The proposed algorithm was coded using C programming language. The experiments were performed on a PC running Linux operating system (Ubuntu 20.04), with 16 GB of RAM memory, and I7 processor (2.3 GHz). A population including 6 search agents was considered and 20 iterations of the algorithm were executed. Results were calculated for three different sizes of the quantized palette: 16, 64 and 256 colors.

Table 1 shows the average results of the errors obtained for each image after applying 20 independent tests, together with the execution time of the tests. The table shows results considering a sampled image with $step = 10$ and an image without sampling ($step = 1$). The supplementary material includes detailed results for each image in the test set, each palette size, and both *step* values.

It can be observed that the execution time is remarkably reduced when sampled images are used to apply GWO+ATCQ. The execution time for the case that considers $step = 10$ is approximately 12% of the time consumed when the original image has not been sampled. On the other hand, the error of the quantized images is very similar in many cases for both *step* values.

5.2. The main operations of the method

Section 5 describes the operations that define the GWO+ATCQ method and includes several figures showing the effect of each operation for a single image (kodim01) and palette size (64 colors). Below are the results obtained for the set of 24 images when each of the fundamental operations of the algorithm is omitted. To do this, the

Table 2
Average MSE, MAE, PSNR, SSIM and execution time (milliseconds) for GWO+ATCQ and several variants.

	colors	BASIC	NO SORT	NO DELTA	NO EQS	NO SAMP	GWO-1	GWO-10
MSE	16	206.9444	207.4816	310.8868	209.4125	204.7712	479.7279	545.7692
	64	61.4082	65.3715	99.2373	65.2294	59.7570	99.0994	99.3712
	256	23.5191	29.3722	34.9268	26.4453	22.1886	30.9865	34.7357
MAE	16	18.0148	18.1479	20.3907	18.3434	18.0064	22.0920	22.6252
	64	9.4518	9.8135	10.9877	9.9715	9.3856	11.0033	10.9959
	256	5.6628	6.2657	6.3826	6.1303	5.5676	6.1467	6.3679
PSNR	16	25.3177	25.3050	23.5308	25.2514	25.3762	23.3060	23.2658
	64	30.6590	30.3113	28.5100	30.3400	30.7902	28.5181	28.5060
	256	34.8226	33.7446	33.0669	34.2729	35.0807	33.6127	33.0889
SSIM	16	0.8914	0.8893	0.8813	0.8885	0.8914	0.8715	0.8698
	64	0.9573	0.9540	0.9472	0.9523	0.9573	0.9470	0.9471
	256	0.9821	0.9779	0.9779	0.9787	0.9821	0.9794	0.9780
T	16	413	404	359	399	3481	3325	334
	64	1437	1328	1231	1347	11 997	11 600	1172
	256	4983	4569	4505	4766	43 647	41 558	4118

method was applied to the set of images by removing one of the following operations:

- sampling of the original image.
- sorting of the initial solutions represented by the search agents.
- improvement of the δ agent by applying ATCQ.
- new equations defined to update the position of each agent.

Twenty independent tests were performed for each variant of the method, image and palette size. Then, the average MSE, MAE, PSNR, SSIM and execution time were calculated for each variant of the method and palette size. The analysis of this information makes it possible to identify the elements of the algorithm that most influence the results.

Table 2 compares the results of the GWO+ATCQ method to those obtained when one of the main operations are eliminated. It includes results for the following cases: the GWO+ATCQ method proposed in this article, as described in Section 5.1 for the case with $step = 10$ (labeled BASIC), GWO+ATCQ without the sorting operation (labeled NO SORT), GWO+ATCQ without the improvement of the δ search agent (labeled NO DELTA), GWO+ATCQ using the original GWO equations to update the position of the search agents (labeled NO EQS), GWO+ATCQ without sampling the original image (labeled NO SAMP). To complete the analysis, the table also shows the results obtained when applying the original GWO algorithm to unsampled images (labeled GWO-1), and to images sampled with $step = 10$ (labeled GWO-10).

Regarding the four error measures, the feature that reduces the error the most is the improvement of the δ search agent. On the other hand, the sampling of the original image with $step = 10$ does not increase the error of the resulting image too much. It is clear that this operation is the one that affects the final result the least, since the average results of the error measures are very similar for the case with sampling (BASIC) and for the case that eliminates the sampling process (NO SAMP). The sorting of the initial solutions increases its influence for the largest palette, while for the other two palettes the effect of the new equations is greater than that of the sorting operation.

The execution time is clearly influenced by the sampling process of the original image. This is the most time-consuming case. On the contrary, the NO DELTA case is the fastest, since ATCQ operations are not applied.

In summary, it is clearly observed that the execution time is mainly affected by the sampling of the original image and the enhancement of the δ agent. When a balance between execution time and quantized image error is required, the sampling of the original image is an option that reduces execution time and also provides good quality images. As for the operation that improves the δ agent, the reduction in execution time obtained by removing that operation does not compensate for the loss of quality of the generated image.

Table 2 also shows the average results for the original GWO method applied to sampled and unsampled images. It is clearly observed that the average error obtained by GWO is always much worse than that obtained by GWO+ATCQ. It is also observed that the NO DELTA variant obtains errors similar to those of GWO for larger palettes. This indicates that the improvement applied to the δ agent in the proposed algorithm is essential for improving the results. Obviously, the execution time of GWO with sampled images is slightly smaller than that of GWO+ATCQ, since the second method includes additional operations. Nevertheless the difference is not very large.

A statistical test was performed to complete the comparison of the cases. Specifically, the Wilcoxon test was applied to compare the results of the basic GWO+ATCQ to those of the 6 cases discussed in the previous paragraphs. The Wilcoxon test can be applied to compare two solution methods and determines if there is no significant difference between the results of both methods. Table 3 shows the test results. To apply the test, five sets of values were considered, corresponding to the mean MSE, MAE, PSNR, SSIM, and execution time of each pair of compared methods. For this reason, the table includes a block with the results corresponding to each of the five cases. For each test, the table shows the sum of the positive (pos) and negative (neg) ranks and the value of the statistic (Z). The probability associated with the value Z is smaller than 0.001 for all cases, so it is not included in the table. The significance level used in the Wilcoxon test was 0.05.

The statistical test indicates that there are significant differences between the results of each pair of methods compared. GWO+ATCQ with $step = 10$ is significantly faster than the case that considers unsampled data (NO SAMP) and the original GWO applied to unsampled data (GWO-1). On the other hand, the result for the four error measures is similar. The GWO+ATCQ method with $step = 10$ is significantly worse than the case with unsampled data (NO SAMP), but significantly better than the remaining variants tested (including the original GWO).

In summary, it is observed that all the operations added to the GWO algorithm to define the GWO+ATCQ method significantly influence the quantization quality. The sampling of the original image is an operation that allows choosing between obtaining a faster solution and obtaining a higher quality solution, which allows the user to decide which of the two options is more important in a particular application.

5.3. Sensitivity analysis

Although the previous results were obtained with the parameter values defined in Section 5.1, the results obtained with other parameter values are summarized in this section. Specifically, several values were considered for the $step$ parameter, the size of the population (N) and the number of iterations of the method ($TMAX$). The values considered for these parameters are the following: $step = \{1, 10, 100, 1000\}$, $TMAX = \{10, 20, 30, 40\}$, $N = \{6, 8, 10, 12\}$.

Table 3
Results of the Wilcoxon signed rank test applied to compare GWO+ATCQ and some variants that do not apply any of its operations (*pos*: sum of positive ranks; *neg*: sum of negative ranks; Z: value of the test statistic). The probability p corresponding to the Z value is < 0.001 for all cases.

	MSE			MAE			PSNR			SSIM			T		
	pos	*neg*	*Z*	*pos*	*neg*	*Z*	*pos*	*neg*	*Z*	*pos*	*neg*	*Z*	*pos*	*neg*	*Z*
NO SORT	224	2404	−6.117	46	2582	−7.116	2419	209	−6.201	2569	59	−7.043	2519	37	−7.111
NO DELTA	0	2628	−7.374	0	2628	−7.374	2628	0	−7.374	2600	28	−7.217	2628	0	−7.374
NO EQS	215	2413	−6.167	11	2617	−7.312	2413	215	−6.167	2560	68	−6.992	2592	36	−7.172
NO SAMP	2603	25	−7.233	2340	288	−5.758	7	2621	−7.334	555	2073	−4.259	0	2628	−7.374
GWO-1	99	2529	−6.818	202	2426	−6.240	2424	204	−6.229	2377	251	−5.965	0	2628	−7.374
GWO-10	0	2628	−7.374	0	2628	−7.374	2628	0	−7.374	2628	0	−7.374	2592	36	−7.172

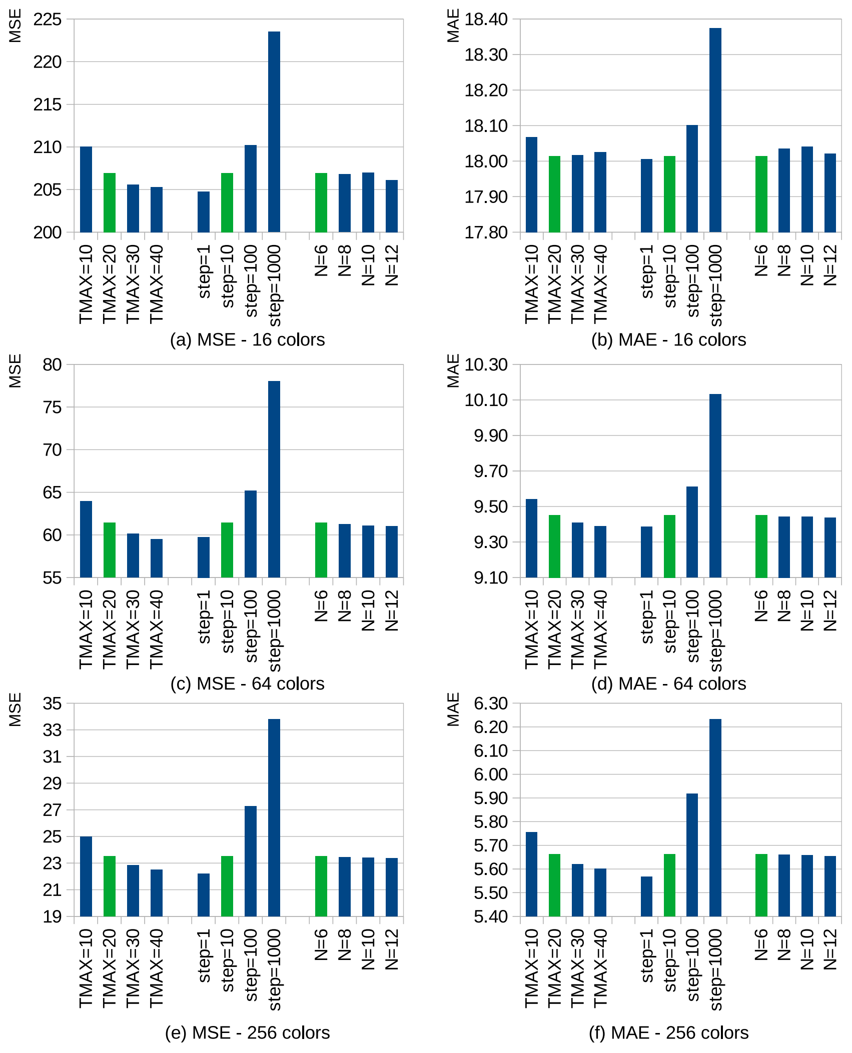

Fig. 8. Average MSE and MAE comparison for GWO+ATCQ results obtained with various parameter values.

Additional tests were carried out modifying one of the three parameters and keeping the other two parameters with the same value proposed in Section 5.1 ($step = 10$, $TMAX = 20$ and $N = 6$). Several figures are included showing the average results obtained for the set of 24 images. In each case, 20 independent tests were performed for each image, palette size and parameter value; then, the average error and the average execution time for the entire set of images were computed. The results are given in Figs. 8–10. A subfigure is included for each palette size, in order to show the results more clearly. To better appreciate the influence of each parameter on the final results, the same subfigure is used to show the results obtained for all the values considered for each of the three parameters. Thus, each subfigure shows three groups of bars. The first group corresponds to the results obtained when the $TMAX$ parameter is modified; the second group shows the results for the $step$ values considered; the last group shows the results for the cases that consider different population sizes. Each group of bars includes a green bar that represents the GWO+ATCQ results obtained with the parameter values proposed in Section 5.1 ($TMAX = 20$, $step = 10$ and $N = 6$). This case is represented with a different color to facilitate its identification in each group of cases.

As expected, it is clearly observed that the error decreases as the iterations of the algorithm increase or the $step$ value decreases. On the other hand, the effect on the execution time is the opposite for both values (when the error decreases, the execution time increases). The $step$ value is the parameter that most influences the execution time. Although using unsampled images generates the best error values, it is clearly observed that this case is much more time-consuming than the cases corresponding to sampled images. Furthermore, the size of the

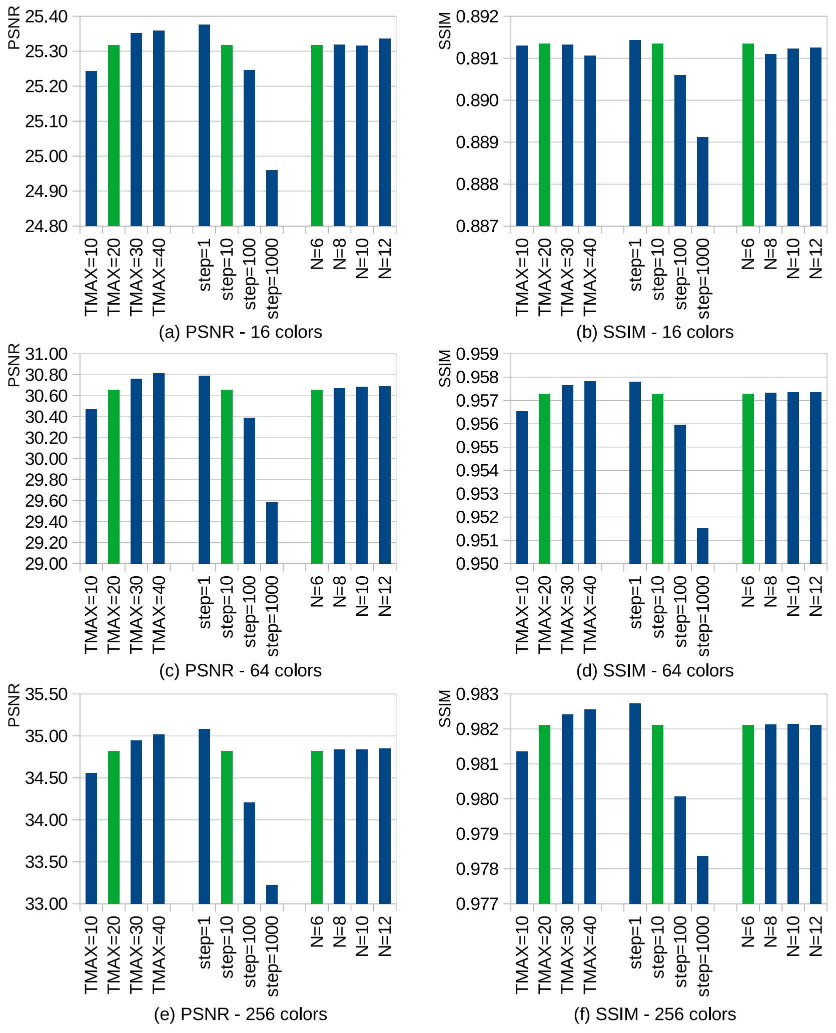

Fig. 9. Average PSNR and SSIM comparison for GWO+ATCQ results obtained with various parameter values.

Table 4
Results of the Wilcoxon signed rank test applied to compare GWO+ATCQ executed with several values of the parameters $TMAX$, $step$ and N (pos: sum of positive ranks; neg: sum of negative ranks; Z: value of the test statistic; p: probability corresponding to the Z value). Since $p < 0.001$ for all the cases that compare T, this information is not included in the table to reduce its size.

	MSE				MAE				PSNR				SSIM				T		
	pos	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*
$TMAX = 10$	24	2604	−7.239	<0.001	170	2458	−6.420	<0.001	2624	4	−7.351	<0.001	2026	602	−3.996	<0.001	2628	0	−7.374
$TMAX = 30$	2523	105	−6.785	<0.001	2156	472	−4.725	<0.001	36	2592	−7.172	<0.001	665	1963	−3.642	<0.001	0	2628	−7.374
$TMAX = 40$	2527	101	−6.807	<0.001	2100	528	−4.411	<0.001	33	2595	−7.189	<0.001	669	1959	−3.620	<0.001	0	2628	−7.374
$step = 1$	2603	25	−7.233	<0.001	2340	288	−5.758	<0.001	7	2621	−7.334	<0.001	555	2073	−4.259	<0.001	0	2628	−7.374
$step = 100$	3	2625	−7.357	<0.001	42	2586	−7.138	<0.001	2625	3	−7.357	<0.001	2427	201	−6.246	<0.001	2628	0	−7.374
$step = 1000$	0	2628	−7.374	<0.001	7	2621	−7.334	<0.001	2628	0	−7.374	<0.001	2580	48	−7.104	<0.001	2628	0	−7.374
$N = 8$	1680	648	−2.054	0.040	1380	1248	−0.370	0.711	830	1798	−2.716	0.007	1221	1407	−0.522	0.602	0	2628	−7.374
$N = 10$	1797	831	−2.710	0.007	1475	1153	−0.903	0.366	730	1898	−3.277	0.001	1200	1428	−0.640	0.522	0	2628	−7.374
$N = 12$	2110	518	−4.467	<0.001	1820	808	−2.840	0.005	411	2217	−5.067	<0.001	1196	1432	−0.662	0.508	0	2628	−7.374

population is the parameter with the least influence on the final error. On the contrary, the execution time depends on this parameter, as more search agents require more operations.

The Wilcoxon test was applied to compare the results obtained in Section 5.1 with $step$=10, $TMAX$=20 and N=6, and those of each of the 9 cases considered in the previous discussion (Table 4). It is observed that there are significant differences for all the cases except 5 (those correponding to SSIM with different N values, and MAE with $N = 8$ and $N = 10$). Therefore, the value of the parameters $TMAX$ and $step$ significantly influences the results obtained by the method, while the parameter N only influences the results obtained for the execution time and the errors MSE and PSNR.

Regarding the error values, the reference case (which uses $step = 10$, $TMAX = 20$ and $N = 6$) is significantly better than the cases with larger $step$, but significantly worse than the case that considers unsampled data ($step = 1$). On the other hand, the reference case is significantly worse than the cases with more iterations. It is also significantly worse than the cases with more search agents when considering the MSE and PSNR results. Regarding the execution time, the results are the opposite to those obtained for the errors. The reference case

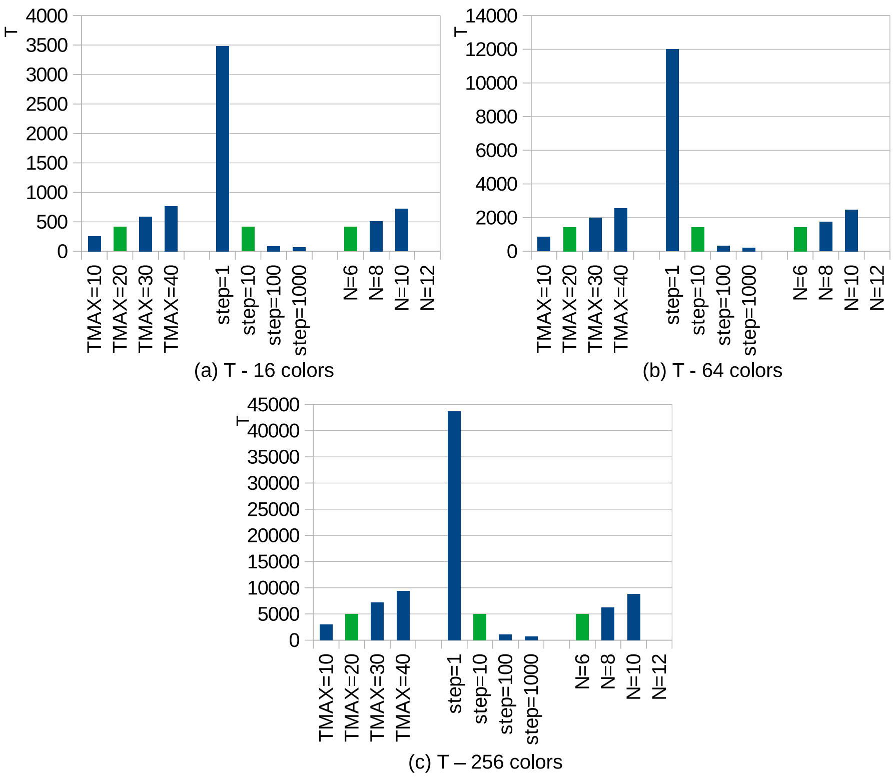

Fig. 10. Average time comparison (milliseconds) for GWO+ATCQ results obtained with various parameter values.

is significantly faster than the one without sampling, but significantly slower than the cases with larger *step*. Furthermore, the reference case is significantly faster than the cases that consider more than 20 iterations and the same is true for the three cases that use more than 6 search agents.

The previous analysis shows that the value of the algorithm parameters influences the final result, both in the quality of the quantized image and in the execution time of the method. The sampling step is the parameter that can reduce the execution time the most, but it also has a great influence on the quality of the resulting image if its value is too large.

5.4. Results of other methods

Some other color quantization methods were tested on the same images, in order to compare their results to those of the proposed method. The methods considered are VB, VC, MC, OC, ADU, BS, NQ, the greedy orthogonal bi-partitioning method (WU) (Wu, 1991), WU combined with ATCQ (WUATCQ) (Pérez-Delgado and Román-Gallego, 2019), BS combined with ATCQ (BSATCQ) (Pérez-Delgado and Román-Gallego, 2020), BS combined with ITATCQ (BSITATCQ) (Pérez-Delgado, 2020b), SFLA applied to color quantization (SFLA-CQ) (Pérez-Delgado, 2019b), PSO combined with ATCQ (PSO +ATCQ) (Pérez-Delgado, 2020a) and ITATCQ. Two optimization algorithms that generate good results, although not specifically designed to solve the color quantization problem, were also applied: adaptive differential evolution with optional external archive (JADE) (Zhang and Sanderson, 2009), and success-history based adaptive differential evolution (SHADE) (Tanabe and Fukunaga, 2013).

Since the tests performed with GWO+ATCQ use a population of 6 search agents, PSO+ATCQ uses 6 particles, SFLA-CQ uses 6 frogs, and SHADE uses a historical memory of size 6. The iterative methods were executed for 20 iterations (the same as GWO+ATCQ). For methods that apply ATCQ and require multiple values of the α parameter, the following values were used: $\alpha = \{0.25, 0.30, 0.35, 0.40, 0.45, 0.50\}$. SFLA-CQ was applied considering the sampling step equal to 10. In addition, 2 memeplexes and 4 improvement iterations were considered for this algorithm. As for the PSO+ATCQ algorithm, the weights used in the equation that calculates the velocity of the particles and the interval that limits the speed were set to the same values proposed in Pérez-Delgado (2020a). ITATCQ was applied considering the modification described in Pérez-Delgado (2021) since it provides better results.

Table 5 summarizes the results obtained with the methods for each palette size considering all the images of the test set. The best value in each results column appears in bold and the second-best appears underlined. The table includes the results for GWO+ATCQ considering both sampling steps. The case with $step = 1$ is labeled GWO+ATCQ-1 and the case with $step = 10$ is labeled GWO+ATCQ-10. The detailed results for each problem in the set are included as supplementary material.

Several conclusions can be drawn from the analysis of the error indices calculated for the methods tested in this article:

- GWO+ATCQ (with $step = 1$ and $step = 10$) outperforms MC, OC, VB, VC, WU, JADE, SHADE, and SFLA-CQ for any palette size and all the error indices. Certainly, MC and OC generate much worse images than all the other methods.

- GWO+ATCQ outperforms NQ and ITATCQ in all cases except one for the set of error measures. For images with 256 colors, NQ obtains better MAE than GWO+ATCQ, while ITATCQ obtains better MSE and PSNR than GWO+ATCQ-10.

- GWO+ATCQ outperforms BS in terms of MSE, PSNR and SSIM, although it obtains worse MAE values for the two larger palettes.

- BSATCQ obtains worse errors than GWO+ATCQ-1 in all cases except MSE and PSNR when considering a 256-color palette. In addition, BSATCQ only obtains better results than GWO+ATCQ-10 for the larger palette and the MSE, PSNR and MAE errors.

Table 5
Average results obtained by several color quantization methods. Results are shown for 16-color, 64-color, and 256-color quantized palettes.

	MSE			MAE			T		
	16	64	256	16	64	256	16	64	256
WU	267.3077	77.0203	27.8067	20.1922	10.6991	6.5445	**1**	**2**	**2**
VB	366.6962	122.4960	47.6271	22.1815	12.3173	7.4339	116	157	199
VC	249.2683	79.2038	34.0929	20.1421	11.3345	7.7528	23	33	53
MC	557.1945	260.8573	140.4371	26.9482	18.3897	13.6371	31	30	30
OC	1026.2450	218.8099	62.9065	40.4983	18.9238	10.2917	56	56	63
ADU	221.5569	61.5356	21.0925	17.9276	9.2370	5.2116	15	39	715
BS	259.5589	73.5253	25.0600	18.3229	8.9307	4.5427	113	189	307
NQ	382.5106	76.5883	24.4757	20.6501	9.5273	5.2758	70	159	275
WUATCQ	220.6627	60.9096	21.0296	**17.2603**	**8.3119**	**4.3764**	16	45	163
BSATCQ	225.0996	61.9886	21.4038	18.7249	9.7571	5.6519	151	304	689
BSITATCQ	214.8677	59.2126	**20.5969**	18.3165	9.5469	5.5611	909	2499	7972
SFLA-CQ	213.0496	64.8728	25.2269	18.0959	9.6072	5.7860	994	3139	14 113
PSO+ATCQ	**200.6647**	**57.7642**	21.2013	17.9357	9.3935	5.5210	7288	15 620	44 549
ITATCQ	272.8545	70.4755	23.3758	21.2611	10.6901	6.1226	932	2064	5776
JADE	236.2184	73.0825	26.7329	18.6830	9.9225	5.8133	6004	15 484	51 054
SHADE	234.9992	72.9159	26.6041	18.5963	9.9473	5.8038	6101	15 687	52 073
GWO+ATCQ-1	204.7712	59.7570	22.1886	18.0064	9.3856	5.5676	3481	11 997	43 647
GWO+ATCQ-10	206.9444	61.4082	23.5191	18.0148	9.4518	5.6628	413	1437	4983

	PSNR			SSIM		
	16	64	256	16	64	256
WU	24.2929	29.7130	34.0381	0.8729	0.9433	0.9741
VB	22.9401	27.8660	31.8427	0.8584	0.9237	0.9591
VC	24.4426	29.4522	33.0513	0.8809	0.9450	0.9667
MC	21.0247	24.3504	27.0072	0.8575	0.9135	0.9436
OC	18.3919	25.0785	30.5118	0.7347	0.8751	0.9434
ADU	24.9987	30.6428	35.3056	**0.8951**	**0.9582**	**0.9838**
BS	24.3684	29.9599	34.6449	0.8755	0.9477	0.9789
NQ	22.6914	29.6637	34.6729	0.8810	0.9558	0.9821
WUATCQ	25.0917	30.7618	35.3584	0.8870	0.9534	0.9801
BSATCQ	24.9661	30.6563	35.2963	0.8856	0.9536	0.9817
BSITATCQ	25.1641	30.8532	**35.4554**	0.8893	0.9554	0.9823
SFLA-CQ	25.1718	30.3697	34.4653	0.8915	0.9565	0.9817
PSO+ATCQ	**25.4641**	**30.9546**	35.2849	0.8915	0.9576	0.9832
ITATCQ	24.2200	30.0717	34.8742	0.8714	0.9461	0.9786
JADE	24.7193	29.8570	34.2535	0.8885	0.9537	0.9807
SHADE	24.7515	29.8718	34.2681	0.8892	0.9533	0.9808
GWO+ATCQ-1	25.3762	30.7902	35.0807	0.8914	0.9578	0.9827
GWO+ATCQ-10	25.3177	30.6590	34.8226	0.8914	0.9573	0.9821

- GWO+ATCQ outperforms BSITATCQ for MSE and PSNR in the 16-color case, and also for MAE in the case of the two smallest palettes. Regarding SSIM, GWO+ATCQ-1 outperforms BSITATCQ and GWO+ATCQ-10 also outperforms it for the two smallest palettes.

- WUATCQ is better than GWO+ATCQ for all palettes with respect to MAE, but is worse with respect to SSIM. In terms of MSE and PSNR, GWO+ATCQ-1 outperforms WUATCQ for the two smallest palettes, but GWO+ATCQ-10 only outperforms WUATCQ for the smallest palette.
- ADU achieves the best SSIM values of the compared methods. It also achieves better MAE values than the proposed method. However, it only achieves better MSE and PSNR than GWO+ATCQ for the 256-color case.
- PSO+ATCQ obtains better average MSE and PSNR values than the proposed method, although the difference in errors is very small for many images. For the other two error indices, PSO+ATCQ only outperforms GWO+ATCQ for two of the palette sizes considered.

It is clear that there is no single method that is best according to all error measures. Considering MSE and PSNR, PSO+ATCQ is the best method for the two smallest palettes, but BSITATCQ is the best for the larger palette. However, WUATCQ is the best method according to MAE, and ADU is the best according to SSIM.

Regarding the worst methods, there is consistency among all the error indices, since the four indicate that MC and OC generate the worst results for all the palettes. Furthermore, WU, VB and VC are also among the worst methods for all error measures, although with better results than MC and OC.

For the 16-color palette, the proposed method is among the top 3 according to MSE and PSNR, and among the top 5 according to MAE and PSNR. For the 64-color palette, the GWO+ATCQ-1 variant ranks between second and fourth best methods (depending on the error considered) while GWO+ATCQ-10 ranks between fourth and sixth. For the 256-color palette, GWO+ATCQ-1 is the third-best method according to SSIM, and it is among the top seven for the other error indices. Meanwhile, GWO+ATCQ-10 is among the top nine methods.

To conclude this analysis, Fig. 11 compares the results of all the methods, but considering the average results of all the palettes. In this case it can be observed that PSO+ATCQ is the only method that obtains better average MSE results than GWO+ATCQ (for both *step* values). PSO+ATCQ and BSITATCQ perform slightly better on PSNR than GWO+ATCQ-1, while WUATCQ, ADU, and BSATCQ also perform better than GWO+ATCQ-10, although the difference among the seven methods is very small. On the other hand, PSO+ATCQ, ADU, BS, and WUATCQ obtain better average MAE than GWO+ATCQ, although the difference between GWO+ATCQ-1 and the first three methods is very small. Finally, PSO+ATCQ and ADU obtain slightly better average SSIM results than the GWO+ATCQ variants.

It can be observed that some of the fastest methods are among those that generate the worst quality images. Fig. 12 compares the average execution time of all the methods discussed in this article. The three slowest methods are JADE, SHADE, and PSO+ATCQ. As expected, it can be observed that splitting-based methods are less time

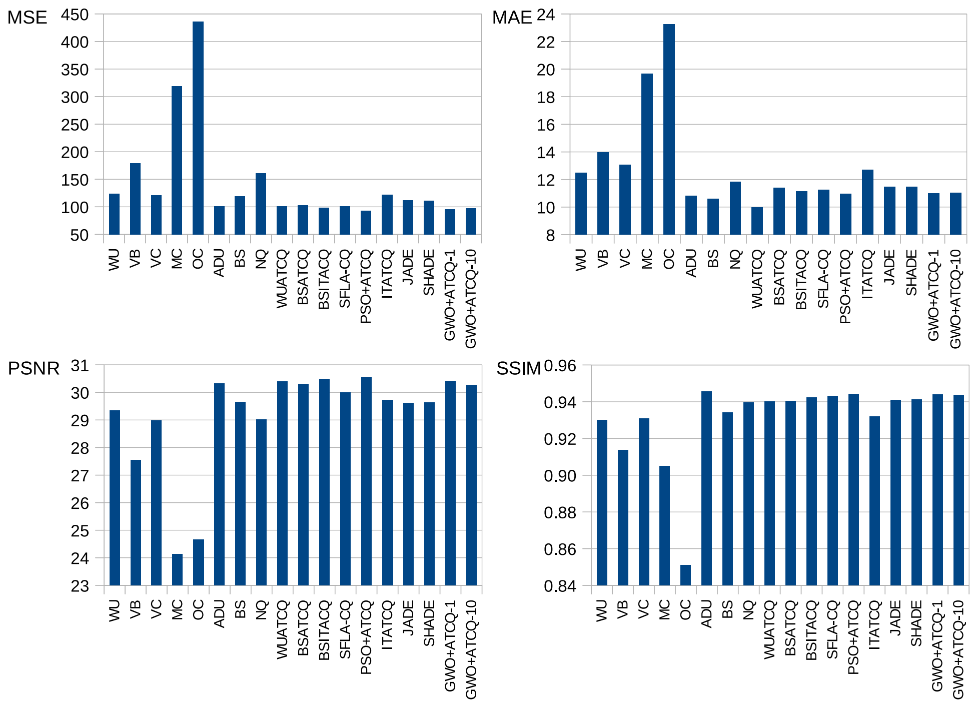

Fig. 11. Average MSE, MAE, PSNR and SSIM comparison for all palettes.

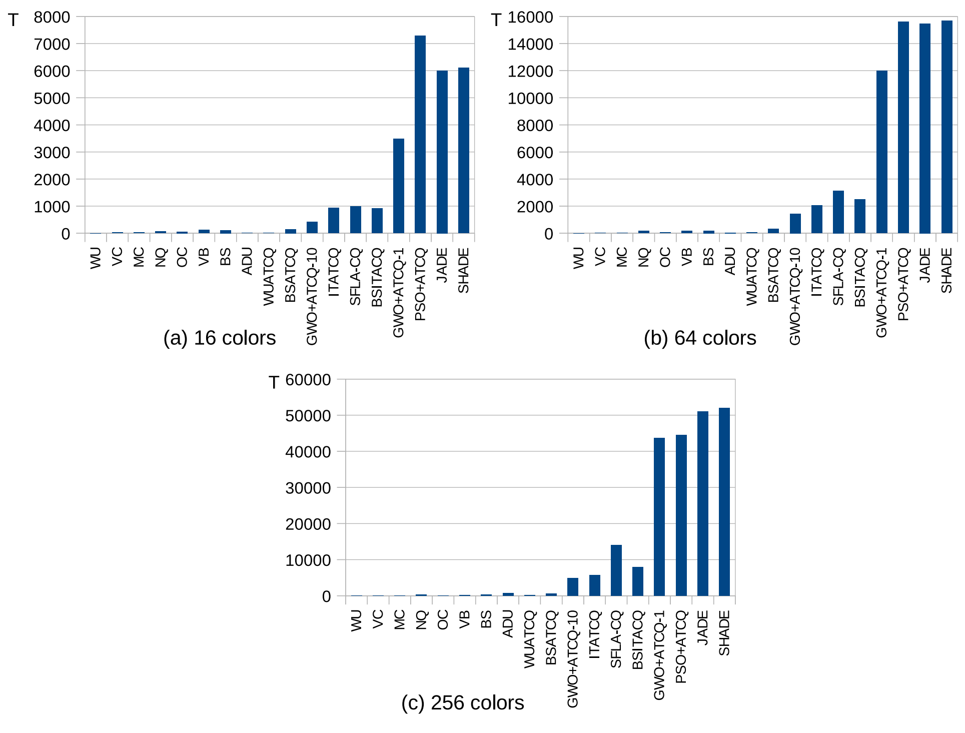

Fig. 12. Average execution time (milliseconds) comparison.

consuming than GWO+ATCQ. On the other hand, GWO+ATCQ applied to unsampled images is faster than the three slowest methods. In addition, the proposed method applied to sampled data is also faster than ITATCQ, BSITATCQ, and SFLA-CQ.

As a summary of the previous discussion, it has been found that the PSO+ATCQ method can obtain better quality images than the proposed method, but it is slower. On the other hand, although GWO+ATCQ is slower than WU, VB, VC, MC, NQ, OC and BSATCQ, it generates better quality images than these methods. ITATCQ, SFLA-CQ and BSITATCQ are faster than the proposed method applied to unsampled images, but they obtain images of poorer quality in many cases. JADE and SHADE are worse than the proposed method in speed and quality of the resulting image. WUATCQ and ADU are faster than the proposed method, generate better quality images regarding MAE, but worse regarding MSE. The PSNR results of both methods are between those of the two GWO+ATCQ variants. Furthermore, ADU generates images with better SSIM values than GWO+ATCQ, while WUATCQ generates images with worse SSIM values.

Table 6
Results of the Wilcoxon signed rank test applied to compare GWO+ATCQ to other color quantization methods (*pos*: sum of positive ranks; *neg*: sum of negative ranks; Z: value of the test statistic; p: probability corresponding to the Z value).

	MSE								MAE							
	GWO+ATCQ-1				GWO+ATCQ-10				GWO+ATCQ-1				GWO+ATCQ-10			
	pos	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*
GWO+ATCQ-10	25	2603	−7.233	<0.001					288	2340	−5.758	<0.001				
WU	0	2628	−7.374	<0.001	3	2625	−7.357	<0.001	0	2628	−7.374	<0.001	0	2628	−7.374	<0.001
VB	0	2628	−7.374	<0.001	3	2625	−7.357	<0.001	0	2628	−7.374	<0.001	0	2628	−7.374	<0.001
VC	0	2628	−7.374	<0.001	3	2625	−7.357	<0.001	0	2628	−7.374	<0.001	0	2628	−7.374	<0.001
MC	0	2628	−7.374	<0.001	3	2625	−7.357	<0.001	0	2628	−7.374	<0.001	0	2628	−7.374	<0.001
OC	0	2628	−7.374	<0.001	3	2625	−7.357	<0.001	0	2628	−7.374	<0.001	0	2628	−7.374	<0.001
ADU	764	1864	−3.086	0.002	1210	1418	−0.584	0.559	2288	340	−5.466	<0.001	2345	283	−5.786	<0.001
BS	9	2619	−7.323	<0.001	61	2567	−7.031	<0.001	2000	628	−3.85	<0.001	2049	579	−4.125	<0.001
NQ	9	2619	−7.323	<0.001	67	2561	−6.998	<0.001	784	1844	−2.974	0.003	918	1710	−2.222	0.026
WUATCQ	761	1867	−3.103	0.002	1168	1460	−0.819	0.413	2616	12	−7.306	<0.001	2614	14	−7.295	<0.001
BSATCQ	436	2192	−4.927	<0.001	822	1806	−2.761	0.006	150	2478	−6.532	<0.001	292	2336	−5.735	<0.001
BSITATCQ	1142	1486	−0.965	0.334	1608	1020	−1.65	0.099	424	2204	−4.994	<0.001	792	1836	−2.929	0.003
SFLA-CQ	0	2628	−7.374	<0.001	57	2571	−7.054	<0.001	173	2455	−6.403	<0.001	337	2291	−5.483	<0.001
PSO+ATCQ	2617	11	−7.312	<0.001	2628	0	−7.374	<0.001	1715	913	−2.25	0.024	2229	399	−5.135	<0.001
ITATCQ	37	2591	−7.166	<0.001	198	2430	−6.263	<0.001	0	2628	−7.374	<0.001	0	2628	−7.374	<0.001
JADE	0	2628	−7.374	<0.001	1	2627	−7.368	<0.001	0	2628	−7.374	<0.001	17	2611	−7.278	<0.001
SHADE	0	2628	−7.374	<0.001	1	2627	−7.368	<0.001	0	2628	−7.374	<0.001	25	2603	−7.233	<0.001

Table 7
Results of the Wilcoxon signed rank test applied to compare GWO+ATCQ to other color quantization methods (*pos*: sum of positive ranks; *neg*: sum of negative ranks; Z: value of the test statistic; p: probability corresponding to the Z value).

	PSNR								SSIM							
	GWO+ATCQ-1				GWO+ATCQ-10				GWO+ATCQ-1				GWO+ATCQ-10			
	pos	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*	*pos*	*neg*	*Z*	*p*
GWO+ATCQ-10	2621	70	−7.334	<0.001					2073	555	−4.259	<0.001				
WU	2628	0	−7.374	<0.001	2626	2	−7.363	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001
VB	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001
VC	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2625	3	−7.357	<0.001
MC	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001
OC	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001
ADU	1633	995	−1.79	0.073	1097	1531	−1.218	0.223	454	2174	−4.826	<0.001	308	2320	−5.645	<0.001
BS	2582	46	−7.116	<0.001	2515	113	−6.74	<0.001	2625	3	−7.357	<0.001	2617	11	−7.312	<0.001
NQ	2600	28	−7.217	<0.001	2553	75	−6.953	<0.001	2298	330	−5.522	<0.001	2140	488	−4.635	<0.001
WUATCQ	1454	1174	−0.786	0.432	998	1630	−1.773	0.076	2519	109	−6.762	<0.001	2472	156	−6.498	<0.001
BSAT	1896	732	−3.266	0.001	1323	1305	−0.051	0.960	2502	126	−6.667	<0.001	2355	273	−5.842	<0.001
BSITATCQ	1087	1541	−1.274	0.203	641	1987	−3.777	<0.001	2159	469	−4.742	<0.001	1901	727	−3.294	<0.001
SFLA	2628	0	−7.374	<0.001	2570	58	−7.048	<0.001	2055	573	−4.158	<0.001	1757	871	−2.486	0.013
PSOATCQ	15	2613	−7.290	<0.001	0	2628	−7.374	<0.001	1174	1454	−0.786	0.432	850	1778	−2.604	0.009
ITATCQ	2573	55	−7.065	<0.001	2357	271	−5.853	<0.001	2628	0	−7.374	<0.001	2628	0	−7.374	<0.001
JADE	2628	0	−7.374	<0.001	2627	1	−7.368	<0.001	2527	101	−6.807	<0.001	2479	149	−6.538	<0.001
SHADE	2628	0	−7.374	<0.001	2626	2	−7.363	<0.001	2452	176	−6.386	<0.001	2401	227	−6.1	<0.001

The Wilcoxon test was performed to complete the comparison of the methods (Tables 6–8). The test was applied to compare the results of GWO+ATCQ with *step* = 1 (labeled GWO+ATCQ-1) and *step* = 10 (labeled GWO+ATCQ-10) to those of the other methods considered in this section. Therefore, each block includes two subsets of columns with results corresponding to both values of *step*. The block set on the left shows the results obtained when GWO+ATCQ-10 is compared to GWO+ATCQ-1. However, the results of the comparison are not repeated in the group of columns on the right side, since the information is redundant. The significance level used in the Wilcoxon test was 0.05.

When the results for the MSE values are considered, it can be observed that there are no significant differences between GWO+ATCQ-1 and BSITATCQ, but the first method is significantly better than the other methods except PSO+ATCQ. When considering GWO+ATCQ-10, PSO+ATCQ is still significantly better than the proposed method. On the other hand, there are no significant differences between GWO+ATCQ-10 and ADU, WUATCQ, or BSITATCQ, and GWO+ATCQ-10 is significantly better than the other methods.

When the MAE results are considered, GWO+ATCQ (with both steps) is significantly better than the other methods except ADU, WUATCQ, BS and PSO+ATCQ.

Regarding PSNR, there are no significant differences between GWO+ATCQ-1 and ADU, WUATCQ or BSITATCQ, and the method is significantly better than the remaining methods except PSO+ATCQ. In addition, there are no significant differences between GWO+ATCQ-10 and ADU, WUATCQ or BSATCQ, and it is significantly better that the other methods except PSO+ATCQ and BSITATCQ.

According to SSIM results, there are no significant differences between GWO+ATCQ-1 and PSO+ATCQ, but the differences are significant for the remaining cases. Only ADU is significantly better than both GWO+ATCQ variants, and PSO+ATCQ is significantly better than GWO+ATCQ-10.

Regarding the execution time, the differences between each pair of methods are significant. GWO+ATCQ-10 is significantly faster than GWO+ATCQ-1, BSITATCQ, SFLA-CQ, ITATCQ, PSO+ATCQ, JADE, and SHADE. On the other hand, GWO+ATCQ-1 is significantly faster than the last three methods in the previous list.

Therefore, the results obtained by applying the Wilcoxon test support the conclusions set forth in the previous discussion.

5.5. Qualitative comparison of methods

The analysis of the numerical results carried out in the previous sections allows us to determine which of the compared methods are better. To try to understand the reasons why one method is faster or

Table 8
Results of the Wilcoxon signed rank test applied to compare GWO+ATCQ to other color quantization methods (*pos*: sum of positive ranks; *neg*: sum of negative ranks; *Z*: value of the test statistic). The probability *p* corresponding to the *Z* value is not included because it is < 0.001 for all cases.

		GWO+ATCQ-1			GWO+ATCQ-10		
		pos	*neg*	*Z*	*pos*	*neg*	*Z*
T	GWO+ATCQ-10	2628	0	−7.374			
	WU	2628	0	−7.374	2628	0	−7.374
	VB	2628	0	−7.374	2628	0	−7.374
	VC	2628	0	−7.374	2628	0	−7.374
	MC	2628	0	−7.374	2628	0	−7.374
	OC	2628	0	−7.374	2628	0	−7.374
	ADU	2628	0	−7.374	2628	0	−7.374
	BS	2628	0	−7.374	2628	0	−7.374
	NQ	2628	0	−7.374	2628	0	−7.374
	WUATCQ	2628	0	−7.374	2628	0	−7.374
	BSATCQ	2628	0	−7.374	2628	0	−7.374
	BSITATCQ	2628	0	−7.374	0	2628	−7.374
	SFLA-CQ	2579	49	−7.099	0	2628	−7.374
	PSO+ATCQ	0	2628	−7.374	0	2628	−7.374
	ITATCQ	2628	0	−7.374	0	2628	−7.374
	JADE	0	2628	−7.374	0	2628	−7.374
	SHADE	0	2628	−7.374	0	2628	−7.374

produces better results than another, the characteristics of each method can be also analyzed. The rest of this section shows this analysis.

Splitting-based color quantization methods are fast because they do not process each individual pixel in the image, but instead divide the solution space into boxes containing pixels and process those boxes. Therefore, methods that apply this solution approach are fast. However, these methods generate a final solution that cannot be improved at the cost of increasing processing time. They are deterministic methods that apply a specific criterion to choose a box and split it, which means that for the same original image the same quantized image is always obtained. On the contrary, clustering-based methods can refine the obtained solution by adjusting the number of iterations that a method applies.

The experimental results show that PSO+ATCQ can produce better quality images, but it takes more time than GWO+ATCQ to obtain a solution. It should be noted that PSO+ATCQ is not only slower than GWO+ATCQ, but also requires more memory. PSO stores three vectors for each search agent in the population, corresponding to the current position, the current velocity and the best position so far of each agent. By contrast, GWO only stores the current position of each search agent. Therefore, PSO+ATCQ requires two additional vectors of size q to perform the color quantization operation. Each iteration of this method should update each agent's position and velocity, plus the personal best position if the current position of the agent has improved. These operations, together with the fact that the fitness of an agent is calculated taking into account all the pixels of the original image, make PSO+ATCQ slower than GWO+ATCQ.

SFLA-CQ works on images sampled using the same technique applied in this article. However, it is observed that the method using frogs is slower. The operation that most influences this result is the number of times the fitness is calculated in each iteration of the algorithm. As indicated when analyzing the complexity of the GWO+ATCQ method, the calculation of the fitness of a position has a high computational cost within the algorithm. In this case, the calculation of the fitness of a position has the same cost both in GWO+ATCQ and in SFLA-CQ, since both methods use sampled images. Each iteration of both algorithms needs to calculate the fitness of all individuals in the population. However, the frog-based method must also compute the fitness for the candidate positions chosen to move the worst agent in each group. This method applies several iterations to improve the position of the worst agent in a group. In the best case, each improvement iteration should calculate a candidate position and its fitness; however, in the worst case, three candidate positions and two fitness values must be calculated. Therefore, although SFLA does not update the position of all the agents in the population, it requires calculating the fitness function more times than GWO.

We can conclude that the fitness calculation is a critical point for the GWO+ATCQ, PSO+ATCQ and SFLA-CQ methods. This operation is the one that requires more calculation time and depends on the size of the set of input pixels. GWO+ATCQ and SFLA-CQ speed up this calculation by working on a reduced set of points. However, it must also be taken into account that the number of times each algorithm performs this operation also has a significant influence on the execution time of the algorithm.

Regarding the ITATCQ method, each iteration of this algorithm processes the n pixels of the original image. However, it does not require computing a global error measure over all the pixels in the image to check the quality of the current quantized palette. Therefore, the speed of this method depends fundamentally on the number of pixels in the image and the size of the quantized palette. An important difference between this method and the other swarm methods used in this article is that ITATCQ does not work with a set of solutions. In PSO+ATCQ, SFLA-CQ and GWO+ATCQ each individual represents a color palette, while in ITATCQ each individual represents a pixel. Therefore, ITATCQ works on a palette that can be improved throughout the iterations, while the other three methods can work on several palettes in parallel and choose the best one. This feature makes it possible for the other methods to define a final quantized palette of better quality.

NQ uses a one-dimensional neural network that is trained using the pixels of the original image. The neuron weights are updated to learn the color distribution of the original image colors. This method can be applied to a sampled image, resulting in a lower quality quantized image, although the process is faster. The results included in this article consider the unsampled image, to obtain the highest quality result that this method can achieve. NQ was proposed to reduce an image to 256 colors, and various authors have verified that it does not work well when applied to quantized palettes that include only a few colors. The numerical results presented in this article also confirm that the best NQ results are obtained for palettes with 256 colors.

ADU applies an iterative process that starts with a single cluster and obtains a total of q clusters by splitting clusters that reach a certain size. As in NQ, each iteration processes all the pixels of the original image. Although both ADU and NQ use competitive learning, it is clearly seen that ADU generates better quality images because it does not impose a one-dimensional structure to the neurons.

WUATCQ, BSATCQ ans BSITATCQ are mixed methods, which apply a clustering method to the solution generated by a splitting-based method. In these cases, the objective is to take advantage of the speed of the first method to obtain a good initial solution and then apply a method that improves the said solution. In general, the longer the second method runs, the better the solution. Therefore, an advantage in applying these methods lies in the possibility of choosing between the speed and the quality of the result (obtain a result quickly or improve that result by taking a little more time). However, it must be taken into account that the palette generated by the first of the two combined methods conditions the final result of the method.

6. Conclusion

This article discusses the application of the grey wolf optimization algorithm to perform color quantization. Several new operations have been added to the original GWO to adapt it to the proposed problem. The method uses the mean squared error as objective function of the color quantization problem, which is formulated as an optimization problem. On the other hand, the input image is sampled to speed up the process. Each search agent in the group represents a quantized palette. The initial palettes associated with the search agents are sorted to allow a better adjustment at the beginning of the algorithm. The equations used to update the position of the search agents include the fitness of the position as a penalty term that improves the result of the algorithm. Furthermore, the position of the δ search agent is improved by applying the ATCQ method. All these features contribute to the algorithm obtaining a good quantized palette to represent the original image.

Computational results show that the proposed method can generate better quality images than most of the compared color quantization methods. It generates better global values for the four error measures analyzed with respect to WU, VB, VC, MC, OC, NQ, ITATCQ, SFLA-CQ, JADE and SHADE methods. On the other hand, it also obtains better global values than other methods, such as BS, BSATCQ, BSITATCQ or WUATCQ, when only some of the four error indices is considered. It has also been proven that sampling the original image allows to speed up the algorithm and still obtain good quality images.

It would be interesting to speed up the proposed method. As has been verified, the execution time of the algorithm mainly depends on the size of the set of pixels that represent the image. Therefore, among the future directions of research, the use of a reduced set of data to apply the method will be analyzed. It should be noted that some of the faster classical methods apply an initial operation in which they create a histogram from the pixels of the original image and then work on that histogram to apply the remaining operations of the method. This same technique could be useful to speed up the proposed method.

As another future direction of research, it is intended to integrate the proposed method into a system to help people with color blindness. This is a visual problem that prevents differentiating certain colors. Among other situations, this problem affects the use of computers and other devices that display images. A useful tool to combat this problem would be a recoloring system that transforms the colors of the palette that a color blind individual confuses into others that the individual can distinguish, thereby allowing this person to recognize contrasts and different chromatic tones. The color quantization method described in this article could be integrated into such a tool to reduce the size of the color palette and identify subsets of pixels in an image whose color is not perceived by the user.

CRediT authorship contribution statement

María-Luisa Pérez-Delgado: Writing – original draft, Validation, Software, Methodology, Funding acquisition, Conceptualization. **Jesús-Ángel Román-Gallego:** Visualization, Software, Formal analysis. **M. Emre Celebi:** Writing – review & editing, Methodology, Conceptualization.

Funding

The work of Dr. Pérez-Delgado and Dr. Román-Gallego was supported by the Samuel Solórzano Barruso Memorial Foundation of the University of Salamanca, Spain [Grant number FS/102015]. Dr. Celebi's work was partially supported by the National Science Foundation, United States under [Award No. OIA-1946391].

Declaration of competing interest

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

Acknowledgments

Dr. Celebi's work was partially supported by the National Science Foundation under Award No. OIA-1946391. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cviu.2026.104659. It includes results for each test image and each method compared to GWO+ATCQ in this article. It also includes figures that compare GWO+ATCQ-1 and GWO+ATCQ-10.

Data availability

The data is available online. The URL is provided in the article.

References

Abernathy, A., Celebi, M.E., 2022. The incremental online k-means clustering algorithm and its application to color quantization. Expert. Syst. Appl. 207, 117927. http://dx.doi.org/10.1016/j.eswa.2022.117927.

Ahmed, H.M., Youssef, B.A., Elkorany, A.S., Saleeb, A.A., Abd El-Samie, F., 2018. Hybrid gray wolf optimizer–artificial neural network classification approach for magnetic resonance brain images. Appl. Opt. 57 (7), B25–B31. http://dx.doi.org/10.1364/AO.57.000B25.

Ali, M., El-Hameed, M., Farahat, M., 2017. Effective parameters' identification for polymer electrolyte membrane fuel cell models using grey wolf optimizer. Renew. Energ. 111, 455–462. http://dx.doi.org/10.1016/j.renene.2017.04.036.

Aljarah, I., Mafarja, M., Heidari, A.A., Faris, H., Mirjalili, S., 2020. Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach. Knowl. Inf. Syst. 62 (2), 507–539. http://dx.doi.org/10.1007/s10115-019-01358-x.

Altan, A., Karasu, S., Zio, E., 2021. A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. Appl. Soft Comput. 100, 106996. http://dx.doi.org/10.1016/j.asoc.2020.106996.

Amirsadri, S., Mousavirad, S.J., Ebrahimpour-Komleh, H., 2018. A levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training. Neural Comput. Appl. 30 (12), 3707–3720. http://dx.doi.org/10.1007/s00521-017-2952-5.

An, N.Y., Pun, C.M., 2014. Color image segmentation using adaptive color quantization and multiresolution texture characterization. Signal, Image Video Process. 8 (5), 943–954.

Berman, D., Avidan, S., et al., 2016. Non-local image dehazing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1674–1682.

Bounds, H., Celebi, M.E., Maxwell, J., 2024. Color quantization using an accelerated Jancey k-means clustering algorithm. J. Electron. Imaging 33 (5), 053052–053052.

Carro-Calvo, L., Salcedo-Sanz, S., Ortiz-García, E.G., Portilla-Figueras, A., 2010. An incremental-encoding evolutionary algorithm for color reduction in images. Integr. Comput.-Aided Eng. 17 (3), 261–269. http://dx.doi.org/10.3233/ICA-2010-0343.

Celebi, M.E., 2009. Fast color quantization using weighted sort-means clustering. J. Opt. Soc. Amer. A 26 (11), 2434–2443. http://dx.doi.org/10.1364/JOSAA.26.002434.

Celebi, M.E., 2011. Improving the performance of k-means for color quantization. Image Vis. Comput. 29 (4), 260–271. http://dx.doi.org/10.1016/j.imavis.2010.10.002.

Celebi, M.E., 2023. Forty years of color quantization: a modern, algorithmic survey. Artif. Intell. Rev. 56 (12), 13953–14034. http://dx.doi.org/10.1007/s10462-023-10406-6.

Celebi, M.E., Hwang, S., Wen, Q., 2014. Colour quantisation using the adaptive distributing units algorithm. Imaging Sci. J. 62 (2), 80–91. http://dx.doi.org/10.1179/1743131X13Y.0000000059.

Celebi, M.E., Kingravi, H.A., Vela, P.A., 2013. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert. Syst. Appl. 40 (1), 200–210. http://dx.doi.org/10.1016/j.eswa.2012.07.021.

Celebi, M.E., Wen, Q., Hwang, S., 2015. An effective real-time color quantization method based on divisive hierarchical clustering. J. Real-Time Image Process. 10 (2), 329–344. http://dx.doi.org/10.1007/s11554-012-0291-4.

Chakraborty, A., Kar, A.K., 2017. Swarm intelligence: A review of algorithms. In: Patnaik, S., Yang, X.S., Nakamatsu, K. (Eds.), Nature-Inspired Computing and Optimization: Theory and Applications. Springer, pp. 475–494.

Chen, X., Kwong, S., Feng, J.F., 2002. A new compression scheme for color-quantized images. IEEE Trans. Circuits Syst. Video Technol. 12 (10), 904–908. http://dx.doi.org/10.1109/TCSVT.2002.804896.

Cheng, M.M., Mitra, N.J., Huang, X., Torr, P.H., Hu, S.M., 2014. Global contrast based salient region detection. IEEE Trans. Pattern Anal. Mach. Intell. 37 (3), 569–582. http://dx.doi.org/10.1109/TPAMI.2014.2345401.

Chou, C.H., Liu, K.C., 2004. Color image compression using adaptive color quantization. In: 2004 International Conference on Image Processing, 2004, Vol. 4. ICIP'04, IEEE, pp. 2331–2334.

Chuang, Y.Y., Curless, B., Salesin, D.H., Szeliski, R., 2001. A bayesian approach to digital matting. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2. CVPR 2001, IEEE, http://dx.doi.org/10.1109/CVPR.2001.990970, II–II.

Dekker, A.H., 1994. Kohonen neural networks for optimal colour quantization. Netw.–Comput. Neural 5 (3), 351–367. http://dx.doi.org/10.1088/0954-898X/5/3/003.

Dewangan, R.K., Shukla, A., Godfrey, W.W., 2019. Three dimensional path planning using grey wolf optimizer for UAVs. Appl. Intell. 49 (6), 2201–2217. http://dx.doi.org/10.1007/s10489-018-1384-y.

Dudani, K., Chudasama, A., 2016. Partial discharge detection in transformer using adaptive grey wolf optimizer based acoustic emission technique. Cogent Eng. 3 (1), 1256083. http://dx.doi.org/10.1080/23311916.2016.1256083.

Emary, E., Zawbaa, H.M., Grosan, C., Hassenian, A.E., 2015. Feature subset selection approach by gray-wolf optimization. In: Afro-European Conference for Industrial Advancement. Springer, pp. 1–13.

Emary, E., Zawbaa, H.M., Hassanien, A.E., 2016. Binary grey wolf optimization approaches for feature selection. Neurocomputing 172, 371–381. http://dx.doi.org/10.1016/j.neucom.2015.06.083.

Franzen, R., 2019. [dataset] Kodak lossless true color image suite. URL: http://r0k.us/graphics/kodak/. (Accessed 10 April 2024).

Freisleben, B., Schrader, A., 1997. Color quantization with a hybrid genetic algorithm. In: 1997 Sixth International Conference on Image Processing and Its Applications, vol. 1, IET, pp. 86–90.

Gervautz, M., Purgathofer, W., 1990. A simple method for color quantization: Octree quantization. In: Glassner, A.S. (Ed.), Graphics Gems. Academic Press Professional, Inc., San Diego, CA, USA, pp. 287–293.

Girgis, M.R., Reda, M.S., 2014. A study of the effect of color quantization schemes for different color spaces on content-based image retrieval. Int. J. Comp. Appl. 96 (12).

González, A.I., Grana, M., Albizuri, F.X., D'Anjou, A., Torrealdea, F.J., 2000. A near real-time Evolution-based Adaptation Strategy for dynamic Color Quantization of image sequences. Inform. Sci. 122 (2–4), 161–183. http://dx.doi.org/10.1016/S0020-0255(99)00119-X.

Guha, D., Roy, P.K., Banerjee, S., 2016. Load frequency control of interconnected power system using grey wolf optimization. Swarm Evol. Comput. 27, 97—115. http://dx.doi.org/10.1016/j.swevo.2015.10.004.

Gupta, S., Deep, K., 2019. A novel random walk grey wolf optimizer. Swarm Evol. Comput. 44, 101–112. http://dx.doi.org/10.1016/j.swevo.2018.01.001.

Hansen, P., Lazić, J., Mladenović, N., 2007. Variable neighbourhood search for colour image quantization. IMA J. Manag. Math. 18 (2), 207–221. http://dx.doi.org/10.1093/imaman/dpm008.

Hassanien, A.E., Emary, E., 2018. Swarm Intelligence: Principles, Advances, and Applications. CRC Press.

Hatta, N., Zain, A.M., Sallehuddin, R., Shayfull, Z., Yusoff, Y., 2019. Recent studies on optimisation method of Grey Wolf Optimiser (GWO): a review (2014–2017). Artif. Intell. Rev. 52 (4), 2651–2683. http://dx.doi.org/10.1007/s10462-018-9634-2.

Heckbert, P., 1982. Color image quantization for frame buffer display. In: Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '82, ACM, New York, NY, USA, pp. 297–307. http://dx.doi.org/10.1145/800064.801294.

Hu, P., Pan, J.S., Chu, S.C., 2020. Improved binary grey wolf optimizer and its application for feature selection. Knowl.-Based Syst. 105746. http://dx.doi.org/10.1016/j.knosys.2020.105746.

Hu, Y.C., Su, B.H., 2008. Accelerated k-means clustering algorithm for colour image quantization. Imaging Sci. J. 56 (1), 29–40. http://dx.doi.org/10.1179/174313107X176298.

Hu, Z., Su, Q., Xia, X., 2016. Multiobjective image color quantization algorithm based on self-adaptive hybrid differential evolution. Comput. Intell. Neurosci. 2016, http://dx.doi.org/10.1155/2016/2450431.

Kamboj, V.K., Bath, S., Dhillon, J., 2016. Solution of non-convex economic load dispatch problem using Grey Wolf Optimizer. Neural Comput. Appl. 27 (5), 1301–1316. http://dx.doi.org/10.1007/s00521-015-1934-8.

Kasuga, H., Yamamoto, H., Okamoto, M., 2000. Color quantization using the fast K-means algorithm. Syst. Comput. Jpn. 31 (8), 33–40. http://dx.doi.org/10.1002/1520-684X(200007)31:8<33::AID-SCJ4>3.0.CO;2-C.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, IEEE, pp. 1942–1948.

Khairuzzaman, A.K.M., Chaudhury, S., 2017. Multilevel thresholding using grey wolf optimizer for image segmentation. Expert. Syst. Appl. 86, 64–76. http://dx.doi.org/10.1016/j.eswa.2017.04.029.

Komaki, G., Kayvanfar, V., 2015. Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. J. Comput. Sci.-Neth. 8, 109–120. http://dx.doi.org/10.1016/j.jocs.2015.03.011.

Kuhn, G.R., Oliveira, M.M., Fernandes, L.A., 2008. An improved contrast enhancing approach for color-to-grayscale mappings. Vis. Comput. 24 (7), 505–514.

Kuo, C.T., Cheng, S.C., 2007. Fusion of color edge detection and color quantization for color image watermarking using principal axes analysis. Pattern Recognit. 40 (12), 3691–3704. http://dx.doi.org/10.1016/j.patcog.2007.03.025.

Li, Q., Chen, H., Huang, H., Zhao, X., Cai, Z., Tong, C., Liu, W., Tian, X., 2017a. An enhanced grey wolf optimization based feature selection wrapped kernel extreme learning machine for medical diagnosis. Comput. Math. Method Med. 2017, http://dx.doi.org/10.1155/2017/9512741.

Li, L., Sun, L., Guo, J., Qi, J., Xu, B., Li, S., 2017b. Modified discrete grey wolf optimizer algorithm for multilevel image thresholding. Comput. Intell. Neurosci. 2017, http://dx.doi.org/10.1155/2017/3295769.

Liu, G.H., Yang, J.Y., Li, Z., 2015. Content-based image retrieval using computational visual attention model. Pattern Recognit. 48 (8), 2554–2566. http://dx.doi.org/10.1016/j.patcog.2015.02.005.

Long, W., Liang, X., Cai, S., Jiao, J., Zhang, W., 2017. A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems. Neural Comput. Appl. 28 (1), 421–438. http://dx.doi.org/10.1007/s00521-016-2357-x.

Losson, O., Macaire, L., 2015. CFA local binary patterns for fast illuminant-invariant color texture classification. J. Real-Time Image Process. 10 (2), 387–401. http://dx.doi.org/10.1007/s11554-012-0302-5.

Lu, C., Gao, L., Li, X., Xiao, S., 2017. A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. Eng. Appl. Artif. Intell. 57, 61–79. http://dx.doi.org/10.1016/j.engappai.2016.10.013.

Malik, M.R.S., Mohideen, E.R., Ali, L., 2015. Weighted distance grey wolf optimizer for global optimization problems. In: 2015 IEEE International Conference on Computational Intelligence and Computing Research. ICCIC, IEEE, pp. 1–6.

Medjahed, S.A., Saadi, T.A., Benyettou, A., Ouali, M., 2016. Gray wolf optimizer for hyperspectral band selection. Appl. Soft Comput. 40, 178–186. http://dx.doi.org/10.1016/j.asoc.2015.09.045.

Miao, D., Chen, W., Zhao, W., Demsas, T., 2020. Parameter estimation of PEM fuel cells employing the hybrid grey wolf optimization method. Energy 193, 116616. http://dx.doi.org/10.1016/j.energy.2019.116616.

Mignotte, M., 2008. Segmentation by fusion of histogram-based k-means clusters in different color spaces. IEEE Trans. Image Process. 17 (5), 780–787. http://dx.doi.org/10.1109/TIP.2008.920761.

Mirjalili, S., 2015. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl. Intell. 43 (1), 150–161. http://dx.doi.org/10.1007/s10489-014-0645-7.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61. http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.

Mittal, N., Singh, U., Sohi, B.S., 2016. Modified grey wolf optimizer for global engineering optimization. Appl. Comput. Intell. Soft Comput. 2016, http://dx.doi.org/10.1155/2016/7950348.

Mosavi, M.R., Khishe, M., Ghamgosar, A., 2016. Classification of sonar data set using neural network trained by Gray Wolf Optimization. Neural Netw. World 26 (4), 393. http://dx.doi.org/10.14311/NNW.2016.26.023.

Muangkote, N., Sunat, K., Chiewchanwattana, S., 2014. An improved grey wolf optimizer for training q-Gaussian Radial Basis Functional-link nets. In: 2014 International Computer Science and Engineering Conference. ICSEC, IEEE, pp. 209–214.

Nolle, L., Schaefer, G., 2007. Colour map design through optimization. Eng. Optim. 39 (3), 327–343. http://dx.doi.org/10.1080/03052150601127958.

Omran, M.G., Engelbrecht, A.P., Salman, A., 2005. A color image quantization algorithm based on particle swarm optimization. Informatica 29 (3).

Orchard, M.T., Bouman, C.A., 1991. Color quantization of images. IEEE Trans. Signal Process. 39 (12), 2677–2690. http://dx.doi.org/10.1109/78.107417.

Ozturk, C., Hancer, E., Karaboga, D., 2014. Color image quantization: a short review and an application with artificial bee colony algorithm. Informatica 25 (3), 485–503.

Pathak, Y., Arya, K., Tiwari, S., 2019. Feature selection for image steganalysis using Levy flight-based grey wolf optimization. Multimedia Tools Appl. 78 (2), 1473–1494. http://dx.doi.org/10.1007/s11042-018-6155-6.

Pérez-Delgado, M.L., 2015. Colour quantization with Ant-tree. Appl. Soft Comput. 36, 656–669. http://dx.doi.org/10.1016/j.asoc.2015.07.048.

Pérez-Delgado, M.L., 2018a. Artificial ants and fireflies can perform colour quantisation. Appl. Soft Comput. 73, 153–177. http://dx.doi.org/10.1016/j.asoc.2018.08.018.

Pérez-Delgado, M.L., 2018b. An iterative method to improve the results of ant-tree algorithm applied to colour quantization. Int. J. Bio-Inspir. Comput. 12 (2), 87–114. http://dx.doi.org/10.1504/IJBIC.2018.094199.

Pérez-Delgado, M.L., 2019a. Color image quantization using the shuffled-frog leaping algorithm. Eng. Appl. Artif. Intell. 79, 142–158. http://dx.doi.org/10.1016/j.engappai.2019.01.002.

Pérez-Delgado, M.L., 2019b. The color quantization problem solved by swarm-based operations. Appl. Intell. 49 (7), 2482–2514. http://dx.doi.org/10.1007/s10489-018-1389-6.

Pérez-Delgado, M.L., 2020a. Color quantization with particle swarm optimization and artificial ants. Soft Comput. 24 (6), 4545–4573. http://dx.doi.org/10.1007/s00500-019-04216-8.

Pérez-Delgado, M.L., 2020b. A mixed method with effective color reduction. Appl. Sci. 10 (21), 7819. http://dx.doi.org/10.3390/app10217819.

Pérez-Delgado, M.L., 2020c. Recent applications of swarm-based algorithms to color quantization. In: Recent Advances on Memetic Algorithms and Its Applications in Image Processing. Springer, pp. 93–118. http://dx.doi.org/10.1007/978-981-15-1362-6_5.

Pérez-Delgado, M.L., 2021. Revisiting the iterative ant-tree for color quantization algorithm. J. Vis. Commun. Image Represent. 103180. http://dx.doi.org/10.1016/j.jvcir.2021.103180.

Pérez-Delgado, M.L., Celebi, M.E., 2024. A comparative study of color quantization methods using various image quality assessment indices. Multimed. Syst. 30 (1), 40. http://dx.doi.org/10.1007/s00530-023-01206-7.

Pérez-Delgado, M.L., Román-Gallego, J.Á., 2019. A hybrid color quantization algorithm that combines the greedy orthogonal bi-partitioning method with artificial ants. IEEE Access 7, 128714–128734. http://dx.doi.org/10.1109/ACCESS.2019.2937934.

Pérez-Delgado, M.L., Román-Gallego, J.-Á., 2020. A two-stage method to improve the quality of quantized images. J. Real-Time Image Process. 17 (3), 581–605. http://dx.doi.org/10.1007/s11554-018-0814-8.

Pérez-Delgado, M.L., Román-Gallego, J.-Á., 2023. Medical image processing by swarm-based methods. In: Role of Data-Intensive Distributed Computing Systems in Designing Data Solutions. Springer, pp. 265–293. http://dx.doi.org/10.1007/978-3-031-15542-0_14.

Phung, S.L., Bouzerdoum, A., Chai, D., 2005. Skin segmentation using color pixel classification: analysis and comparison. IEEE Trans. Pattern Anal. Mach. Intell. 27 (1), 148–154. http://dx.doi.org/10.1109/TPAMI.2005.17.

Ponti, M., Nazaré, T.S., Thumé, G.S., 2016. Image quantization as a dimensionality reduction procedure in color and texture feature extraction. Neurocomputing 173, 385–396.

Precup, R.E., David, R.C., Petriu, E.M., 2016. Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity. IEEE Trans. Ind. Electron. 64 (1), 527–534. http://dx.doi.org/10.1109/TIE.2016.2607698.

Roberto e Souza, M., Carlos Sousa e Santos, A., Pedrini, H., 2020. A hybrid approach using the k-means and genetic algorithms for image color quantization. In: De, S., Dey, S., Bhattacharyya, S. (Eds.), Recent Advances in Hybrid Metaheuristics for Data Clustering. Wiley Online Library, pp. 151–171.

Rodríguez, L., Castillo, O., Soria, J., Melin, P., Valdez, F., Gonzalez, C.I., Martinez, G.E., Soto, J., 2017. A fuzzy hierarchical operator in the grey wolf optimizer algorithm. Appl. Soft Comput. 57, 315–328. http://dx.doi.org/10.1016/j.asoc.2017.03.048.

Schaefer, G., Agarwal, P., Celebi, M.E., 2017. Effective colour reduction using grey wolf optimisation. In: European Congress on Computational Methods in Applied Sciences and Engineering. Springer, pp. 170–178.

Schaefer, G., Nolle, L., 2006. A hybrid differential evolution approach to colour map generation. In: Proceedings 20th European Conference on Modelling and Simulation. pp. 434–436.

Schaefer, G., Nolle, L., 2015. A hybrid color quantization algorithm incorporating a human visual perception model. Comput. Intell. 31 (4), 684–698. http://dx.doi.org/10.1111/coin.12043.

Scheunders, P., 1997. A comparison of clustering algorithms applied to color image quantization. Pattern Recogn. Lett. 18 (11–13), 1379–1384. http://dx.doi.org/10.1016/S0167-8655(97)00116-5.

Shakarami, M.R., Davoudkhani, I.F., 2016. Wide-area power system stabilizer design based on grey wolf optimization algorithm considering the time delay. Electr. Power Syst. Res. 133, 149–159. http://dx.doi.org/10.1016/j.epsr.2015.12.019.

Shankar, K., Lakshmanaprabu, S., Khanna, A., Tanwar, S., Rodrigues, J.J., Roy, N.R., 2019. Alzheimer detection using Group Grey Wolf Optimization based features with convolutional classifier. Comput. Electr. Eng. 77, 230–243. http://dx.doi.org/10.1016/j.compeleceng.2019.06.001.

Sharma, P., Sundaram, S., Sharma, M., Sharma, A., Gupta, D., 2019. Diagnosis of Parkinson's disease using modified grey wolf optimization. Cogn. Syst. Res. 54, 100–115. http://dx.doi.org/10.1016/j.cogsys.2018.12.002.

Song, X., Tang, L., Zhao, S., Zhang, X., Li, L., Huang, J., Cai, W., 2015. Grey wolf optimizer for parameter estimation in surface waves. Soil Dyn. Earthq. Eng. 75, 147–157. http://dx.doi.org/10.1016/j.soildyn.2015.04.004.

Su, Q., Hu, Z., 2013. Color image quantization algorithm based on self-adaptive differential evolution. Comput. Intell. Neurosci. 2013, http://dx.doi.org/10.1155/2013/231916, 3–3.

Sulaiman, M.H., Mustaffa, Z., Mohamed, M.R., Aliman, O., 2015. Using the gray wolf optimizer for solving optimal reactive power dispatch problem. Appl. Soft Comput. 32, 286–292. http://dx.doi.org/10.1016/j.asoc.2015.03.041.

Sun, X., Hu, C., Lei, G., Guo, Y., Zhu, J., 2019. State feedback control for a PM hub motor based on gray wolf optimization algorithm. IEEE Trans. Power Electr. 35 (1), 1136–1146. http://dx.doi.org/10.1109/TPEL.2019.2923726.

Tanabe, R., Fukunaga, A., 2013. Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation. IEEE, pp. 71–78.

Tang, J., Liu, G., Pan, Q., 2021. A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. IEEE/CAA J. Autom. Sin. 8 (10), 1627–1643. http://dx.doi.org/10.1109/JAS.2021.1004129.

Taşdizen, T., Akarun, L., Ersoy, C., 1998. Color quantization with genetic algorithms. Signal Process.-Image 12 (1), 49–57. http://dx.doi.org/10.1016/S0923-5965(97)00035-0.

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

Tikhamarine, Y., Souag-Gamane, D., Ahmed, A.N., Kisi, O., El-Shafie, A., 2020. Improving artificial intelligence models accuracy for monthly streamflow forecasting using grey Wolf optimization (GWO) algorithm. J. Hydrol. 582, 124435. http://dx.doi.org/10.1016/j.jhydrol.2019.124435.

Tripathi, A.K., Sharma, K., Bala, M., 2018. A novel clustering method using enhanced grey wolf optimizer and mapreduce. Big Data Res. 14, 93–100. http://dx.doi.org/10.1016/j.bdr.2018.05.002.

Tsai, P., Hu, Y.C., Chang, C.C., 2004. A color image watermarking scheme based on color quantization. Signal Process. 84 (1), 95–106. http://dx.doi.org/10.1016/j.sigpro.2003.07.012.

Vosooghifard, M., Ebrahimpour, H., 2015. Applying grey wolf optimizer-based decision tree classifer for cancer classification on gene expression data. In: 2015 5th International Conference on Computer and Knowledge Engineering. ICCKE, IEEE, pp. 147–151.

Wan, S., Prusinkiewicz, P., Wong, S., 1990. Variance-based color image quantization for frame buffer display. Color Res. Appl. 15 (1), 52–58. http://dx.doi.org/10.1002/col.5080150109.

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

Wu, X., 1992. Color quantization by dynamic programming and principal analysis. ACM Trans. Graph. 11 (4), 348–372.

Zhang, J., Sanderson, A.C., 2009. JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13 (5), 945–958. http://dx.doi.org/10.1109/TEVC.2009.2014613.

Zhang, S., Zhou, Y., 2015. Grey wolf optimizer based on Powell local optimization method for clustering analysis. Discrete Dyn. Nat. Soc. 2015, http://dx.doi.org/10.1155/2015/481360.

Zhang, S., Zhou, Y., Li, Z., Pan, W., 2016. Grey wolf optimizer for unmanned combat aerial vehicle path planning. Adv. Eng. Softw. 99, 121–136. http://dx.doi.org/10.1016/j.advengsoft.2016.05.015.

Zhao, X., Zhang, X., Cai, Z., Tian, X., Wang, X., Huang, Y., Chen, H., Hu, L., 2019. Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients. Comput. Biol. Chem. 78, 481–490. http://dx.doi.org/10.1016/j.compbiolchem.2018.11.017.