IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2021–04–23.
Submitted on 2021–04–06, accepted on 2021–04–20.
ISSN 2105–1232 © 2021 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2021.344

Automatic 1D Histogram Segmentation and Application to the Computation of Color Palettes

Jose-Luis Lisani, Ana Belén Petro

Universitat Illes Balears, IAC3-IEEC (Spain)
joseluis.lisani, anabelen.petro@uib.es

Communicated by Luis Álvarez *Demo edited by* Jose-Luis Lisani

Abstract

This article presents an implementation of the FTC (Fine-to-Coarse) algorithm for histogram segmentation, presented by Delon et al. in 2007. This algorithm uses a non-parametric a contrario approach to segment a 1D histogram into its meaningful modes. We describe also how the method may be applied to the hue, saturation and intensity histograms of color images in order to automatically extract their more representative colors, the so-called color palette. The algorithm for color palette extraction described in this paper is based on the one first published in 2007 by Delon et al., with an improvement that affects low-saturated colors. Several results illustrate the effectiveness of the algorithm.

Source Code

The C source codes implementing the algorithms described in the paper, and the online demo, are accessible at the associated web page[1].

Keywords: 1D histogram segmentation; a contrario method; color palette

1 Introduction

Histograms are extensively used in data analysis because they provide a compact representation of large amounts of data. Moreover, it is often possible to infer global properties of the data from the behavior of their histogram. Histograms can be described by the list of their modes, i.e., the ranges of values around which the data are concentrated.

In 2007 Delon et al. [8] proposed a non-parametric algorithm (the *FTC algorithm*) for the segmentation of a 1D histogram into its meaningful modes. The *meaningfulness* of each node was determined using the a contrario approach, following previous works of the same authors [9, 10, 11]. The algorithm was later refined in [3].

[1] https://doi.org/10.5201/ipol.2021.344

The FTC algorithm has found applications in diverse problems, such as camera stabilization [6], video demultiplexing [13], color palette extraction [7], and satellite image analysis [1].

In this paper we provide a detailed description of the FTC algorithm, and we illustrate its application to the extraction of the main colors of a digital image, as proposed in [7] and [3].

The paper is organized as follows: the histogram segmentation algorithm is described in Section 2, the algorithm for color palette extraction is presented in Section 3, several experiments are shown in Section 4, and, finally, some conclusions are summarized in Section 5.

2 Automatic Location of the Meaningful Modes of a Histogram

As commented in the Introduction, a *mode* of a histogram can be loosely defined as a range of values in which data concentrate. Our goal will be to locate these modes in the 1D case.

More formally, we seek to find intervals of values (which we call *segments*) on which it is "likely" that the histogram is the realization of a **unimodal law**. A density function f is said to be unimodal on some interval $[a, b]$ if f is increasing on some $[a, c]$ and decreasing on $[c, b]$, for some c in (a, b).

Such a segmentation is usually not unique (think of the segmentation defined by all the local minima of the histogram) and we will impose two conditions that lead to a unique result:

- in each segment, the histogram is "statistically unimodal",
- no union of consecutive segments is "statistically unimodal".

The notion of "statistically unimodal" histogram is presented next.

2.1 Testing a Histogram against a given Probability Law

Consider a discrete histogram $h = (h_i)_{i=1\dots L}$, with N samples on L bins $\{1, \dots L\}$. The number h_i is the value of h in bin i and we have that

$$\sum_{i=1}^{L} h_i = N. \tag{1}$$

For each discrete interval $[a, b]$ of $\{1, \dots L\}$, let $h(a, b)$ denote the proportion of points in $[a, b]$,

$$h(a, b) = \frac{1}{N} \left(\sum_{i=a}^{b} h_i \right). \tag{2}$$

Consider now a discrete probability law $p = (p_i)_{i=1\dots L}$. For each interval $[a, b]$ of $\{1, \dots L\}$, let $p(a, b)$ be the probability for a point to fall into the interval $[a, b]$,

$$p(a, b) = \sum_{i=a}^{b} p_i. \tag{3}$$

We define the **relative entropy** of an interval $[a, b]$ of h with respect to p as

$$H_{h,p}([a, b]) = h(a, b) \log \frac{h(a, b)}{p(a, b)} + (1 - h(a, b)) \log \frac{1 - h(a, b)}{1 - p(a, b)} \tag{4}$$

$H_{h,p}$ gives a measure of how well h follows the law p in a given interval[2]. If the relative entropy is large we can state that h doesn't follow the law p. In particular, we say that the interval $[a, b]$ is a ε**-meaningful rejection** of law p if

$$H_{h,p}([a,b]) \geq \frac{1}{N} \log \frac{L(L+1)}{2\varepsilon} \tag{5}$$

In [3] it is proved that the expected number of intervals $[a, b]$ of h that satisfy the previous condition is less than ε when h is indeed following law p in $[a, b]$. In general ε is set to 1.

2.2 Testing the Unimodal Hypothesis

As commented above, an unimodal law can be described as the concatenation of an increasing function with a decreasing function. More precisely, we say that a histogram h **follows the unimodal hypothesis** on the interval $[a, b]$ if there exists $c \in (a, b)$ such that h follows the increasing hypothesis on $[a, c]$ and h follows the decreasing hypothesis on $[c, b]$.

Moreover, we say that a histogram h **follows the decreasing hypothesis** (resp. the increasing hypothesis) on an interval $[a, b]$ if the restriction of the histogram to $[a, b]$ (i.e. $h_{|[a,b]} = (h_a, h_{a+1}, \dots h_b)$) contains no ε-meaningful rejection for the decreasing (resp. increasing) hypothesis. In particular, we compute

$$C_{[a,b]} = N \cdot H - \log \frac{L(L+1)}{2\varepsilon} \tag{6}$$

with H the maximum of the relative entropies of all the subintervals of $[a, b]$ with respect to the hypothesis. If $C_{[a,b]} \geq 0$ the hypothesis is rejected.

The decreasing (resp. increasing) laws against which the intervals of h must be tested can be computed from h itself using the Grenander estimator [12], as explained next.

Let $\mathcal{P}(L)$ be the space of discrete probability distributions on $\{1, \dots L\}$, i.e., the vectors $r = (r_i)_{i=1,\dots L}$ such that

$$\forall i \in \{1, 2, \dots, L\}, \qquad r_i \geq 0 \qquad \text{and} \qquad \sum_{i=1}^{L} r_i = 1. \tag{7}$$

Let $\mathcal{D}(L) \subset \mathcal{P}(L)$ be the space of all decreasing densities on $\{1, \dots L\}$. If $r = \frac{1}{N} h \in \mathcal{P}(L)$ is the normalized histogram of our observations, the decreasing Grenander estimator of r, denoted by $\bar{r}$, is the only distribution of $\mathcal{D}(L)$ which achieves the minimal Kullback-Leibler distance from r to $\mathcal{D}(L)$.

This estimator can be computed using the **Pool Adjacent Violators** algorithm (see [2] and [4]), which is summarized as follows:

1. *Consider the operator* $D : \mathcal{P}(L) \to \mathcal{P}(L)$, *applied on* $r = (r_i)_{i=1,\dots L} \in \mathcal{P}(L)$,
2. *For each interval* $[i, j]$ *on which* r *is increasing,* i.e. $r_i \leq r_{i+1} \leq \dots \leq r_j$ *and* $r_{i-1} > r_i$ *and* $r_{j+1} < r_j$, *define* D *as*

$$D(r)_k = \frac{r_i + \dots + r_j}{j - i + 1} \qquad \text{for} \qquad k \in [i, j]. \tag{8}$$

3. *Otherwise, define* D *as* $D(r)_k = r_k$

This operator D replaces each increasing part of r by a constant value (equal to the mean value on the interval).

A similar procedure can be used to compute the increasing Grenander estimator. A detailed description of the method is provided in Algorithm 3.

[2]More precisely, $H_{h,p}([a, b])$ is the Kullback-Leibler distance between the two Bernoulli distributions of respective parameters h(a, b) and p(a, b) (see [5]).

2.3 The FTC Algorithm

The two conditions stated at the beginning of this section for obtaining an adequate segmentation of a histogram into its modes can be fulfilled by applying the following algorithm, known as Fine to Coarse (or FTC) algorithm:

1. *Define the initial segmentation, $S = \{s_0, \ldots, s_n\}$, as the finest segmentation given by the list of all the minima, plus the end points 1 and L of the histogram.*

2. *Repeat:*

   (a) *For each i, $i = 0, \ldots, n-2$, the cost of the union of interval $[s_i, s_{i+1}]$ with its successive interval $[s_{i+1}, s_{i+2}]$ is defined as the lowest rejection against the unimodal hypothesis on their union*

   $$C_i = C([s_i, s_{i+1}] \cup [s_{i+1}, s_{i+2}]) = \min_{[a,b] \in [s_i, s_{i+1}] \cup [s_{i+1}, s_{i+2}]} C_{[a,b]},$$

   with $C_{[a,b]}$ as defined in Equation (6).
   Obtain the list $L = \{C_0, \ldots, C_{n-2}\}$.

   (b) *Sort the list of costs L, in increasing order.*

   (c) *Assume that the lower cost is C_j. If C_j is lower than 0, then merge the intervals $[s_j, s_{j+1}]$ and $[s_{j+1}, s_{j+2}]$.*

   (d) *Update S.*

   Stop when no more pair of successive intervals follows the unimodal hypothesis, that is, when the lower cost is higher than 0.

3. *Repeat step 2 with the unions of j segments, j going from 3 to length(S).*

A detailed description of the method is provided in Algorithms 1 and 2.

Adaptation of FTC to circular histograms. If $h = (h_i)_{i=1\ldots L}$ is a circular histogram (e.g. a histogram of angular values), the previous algorithm can be adapted easily by defining an auxiliary histogram $\tilde{h}$ as the concatenation of three copies of h. The FTC algorithm is applied to $\tilde{h}$ but only the modes contained in the central part of this new histogram are used as output. The process is described in Algorithm 4.

Algorithm 1: FTC algorithm

Input : input histogram, $h = (h_i)_{i=1 \cdots L}$
Input : maximum expected number of meaningful rejections of unimodal law, ε (default: $\varepsilon = 1$)
Output : segmentation of the histogram into n meaningful modes, $s = \{s_0, \cdots, s_n\}$, with $1 = s_0 < s_1 < \cdots < s_n = L$

```
//Compute local minima of the histogram (including endpoints)
m = {m_1, ..., m_K} = GetMinima(h)          //K values, with m_1 = 1 and m_K = L
//Compute local maxima of the histogram
M = {M_1, ..., M_{K-1}} = GetMaxima(h)      //K-1 values, we have that m_i < M_i < m_{i+1}
//Initial segmentation:  s = m
//Each interval I_i = [m_i, m_{i+1}] follows the unimodal law
//([m_i, M_i] is monotonically increasing and [M_i, m_{i+1}] is monotonically decreasing)
//The number of intervals is K - 1
//Merging of intervals
J = 1                                       //Number of consecutive intervals to merge= 1 + J
while J < K - 1 do
    do
        //Compute cost of merging 1 + J consecutive intervals
        //from I_i = [m_i, m_{i+1}] to I_{i+J} = [m_{i+J}, m_{i+J+1}]
        C ← ∅                                                          //List of costs
        for i = {1, ..., K - J - 1} do
            //Compute cost of considering [m_i, M_{i+J}] monotonically increasing
            cost_I = CostMonotone(h, m_i, M_{i+J}, 'increasing', ε)          //Algorithm 2
            //Compute cost of considering [M_i, m_{i+J+1}] monotonically decreasing
            cost_D = CostMonotone(h, M_i, m_{i+J+1}, 'decreasing', ε)        //Algorithm 2
            //Final cost is the smallest of both values
            //Store index, cost and type
            if cost_I < cost_D then
                C_i = {i, cost_I, 'increasing'}
            else
                C_i = {i, cost_D, 'decreasing'}
            C ← C ∪ C_i                                                //Add cost to list
        //Find minimum of all costs
        {i*, mincost, type} = C* ∈ C such that C*(cost) < C_j(cost)   ∀C_j ∈ C
        //Merge intervals while no meaningful rejection to monotone hypothesis
        if mincost < 0 then
            Update list of minima m: remove from m_{i*+1} to m_{i*+J}.
            if type == 'increasing' then
                Update list of maxima M: remove from M_{i*} to M_{i*+J-1}.
            else
                Update list of maxima M: remove from M_{i*+1} to M_{i*+J}.
    while mincost < 0
    //Increase number of intervals to merge
    J ← J + 1
s = m                                       //Return final list of minima (including end points)
return s
```

Algorithm 2: CostMonotone function

Input : input histogram, $h = (h_i)_{i=1 \cdots L}$
Input : endpoints of an interval of the histogram, i_1, i_2
Input : type of computation, *type* $\in$ 'increasing', 'decreasing'
Input : maximum expected number of meaningful rejections of unimodal law, ε
Output : Cost of considering interval $[i_1, i_2]$ monotonically increasing or decreasing

//The cost is computed w.r.t. a monotonically increasing or decreasing version of the histogram in $[i_1, i_2]$, which is computed using the Pool Adjacent Violators algorithm

//Get subhistogram
$h^s = \{h_i \text{ such that } i \in \{i_1, \cdots, i_2\}\}$
//Number of bins:
$L = i_2 - i_1 + 1$
//Number of samples in subhistogram:
$N = \sum_{i=1}^{L} h_i^s$
//Get monotone version of subhistogram (Grenander estimator of h^s)
$\bar{r} = PoolAdjacentViolators(h_s, type)$ //Algorithm 3
//Normalize histograms
$h^s \leftarrow \frac{h^s}{N}$
$\bar{r} \leftarrow \frac{\bar{r}}{N}$

//Compute maximum relative entropy between all possible intervals of h^s and $\bar{r}$
$H = \max_{[a,b] \in \{1, \cdots, L\}} H_{h^s, \bar{r}}([a, b])$ //$H_{h^s, \bar{r}}([a, b])$ is defined in Equation (4)

//Compute cost C
//C assesses the expected number of rejections of the monotone hypothesis
//If $C \geq 0$ the monotone hypothesis is meaningfully rejected
//Smaller costs mean that the monotone hypothesis is less meaningfully rejected
$C = N \cdot H - \log(L(L+1)/2\varepsilon)$ //Equation (6)

return C

Algorithm 3: Pool Adjacent Violators Algorithm

Input : input histogram, $h = (h_i)_{i=1\cdots L}$
Input : type of computation, *type* $\in$ 'increasing', 'decreasing'
Output : output monotone histogram, $\bar{r} = (r_i)_{i=1\cdots L}$

`//Initialize output histogram`
$\bar{r} = h$
do
  if *type* == *'increasing'* **then**
    `//List of monotonically decreasing intervals`
    $\mathcal{I} \leftarrow$ Intervals $[i, j]$ such that $r_i \geq r_{i+1} \geq \cdots \geq r_j$ and $r_i > r_{i-1}$ and $r_j < r_{j+1}$
  else
    `//List of monotonically increasing intervals`
    $\mathcal{I} \leftarrow$ Intervals $[i, j]$ such that $r_i \leq r_{i+1} \leq \cdots \leq r_j$ and $r_i < r_{i-1}$ and $r_j > r_{j+1}$
  `//Replace each monotonically decreasing/increasing interval by constant value`
  for *each* $[i, j] \in \mathcal{I}$ **do**
    for $k = \{i, \cdots, j\}$ **do**
      $r_k = \frac{r_i + \cdots + r_j}{j - i + 1}$
while $\mathcal{I}$ *is not empty*
return $\bar{r}$

Algorithm 4: FTC algorithm (for circular histograms)

Input : input circular histogram, $h = (h_i)_{i=1\cdots L}$
Input : maximum expected number of meaningful rejections of unimodal law, ε (default: $\varepsilon = 1$)
Output : segmentation of the histogram into $n + 1$ meaningful modes, $s = \{s_0, \cdots, s_n\}$, with $s_0 < s_1 < \cdots < s_n$, (last mode in $[s_n, s_0]$)

`//Build auxiliary histogram: concatenation of 3 copies of input histogram`
$h_{\text{aux}} = \{h, h, h\}$
`//Segmentation of auxiliary histogram`
$s_{\text{aux}} =$ `FTC_algorithm`$(h_{\text{aux}}, \varepsilon)$
`//Segmentation of input histogram`
$s \longleftarrow \emptyset$ `//Initialize output to empty set`
for *each* $s_i \in s_{aux}$ **do**
  if $s_i > L$ ***and*** $s_i \leq 2L$ **then**
    $s \longleftarrow s \cup \{s_i - L\}$ `//Add shifted value to final list`
return s

2.4 Examples

Figure 1-top displays two examples of the use of the FTC algorithm for the segmentation of a histogram into its modes, using the default parameter $\varepsilon = 1$. The histograms in both images are identical, but the one on the right has been segmented under the assumption that it is circular (i.e. using Algorithm 4). In the bottom row the histogram has been segmented using a smaller value of ε ($\varepsilon = 0.001$), which leads to a coarser segmentation.

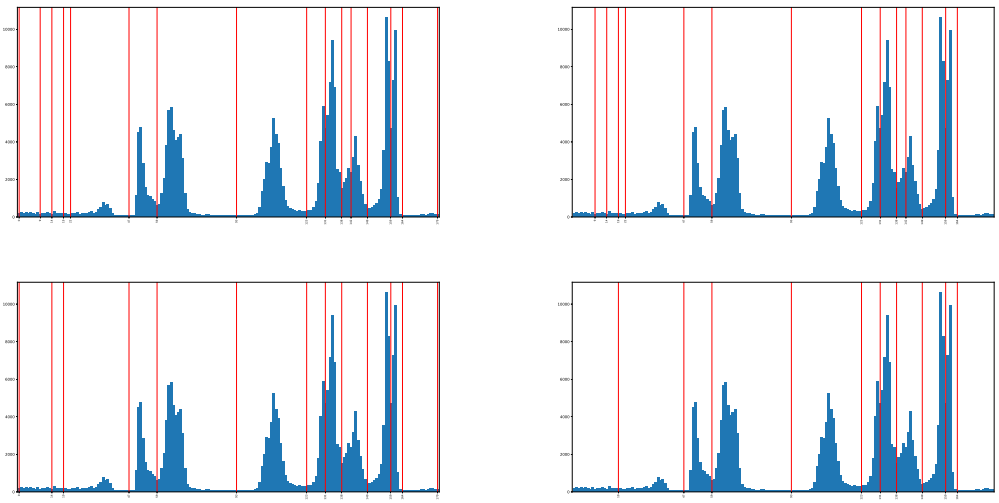


Figure 1: Top, segmentation of a histogram into its modes using the "classical" (left) and "circular" (right) versions of the FTC algorithm, with $\varepsilon = 1$. Bottom, results with $\varepsilon = 0.001$.

3 Color Palette Estimation

In this section we describe the application of the FTC algorithm to the analysis of the colors of an image. In particular, we deal with the problem of finding the minimum set of colors needed to describe an image. By analogy between this set of colors and the palette of a painter the problem is usually referred to as *color palette estimation*.

The method presented in this section was proposed in [7] (refer also to [3] for further details). The authors imposed two requirements for the construction of their palette, namely, reduction of redundant colors, and preservation of rare colors. They proposed a hierarchical algorithm in which colors were first discriminated by their hue; next, colors with similar hue were discriminated by their saturation; and finally, colors with similar hue and saturation were discriminated by their intensity. At each step of the algorithm the discrimination between different values of a given magnitude (hue, saturation or illumination) was performed by analyzing the associated 1D histograms using the FTC algorithm.

3.1 Color Representation

The color palette is estimated from the HSI representation of the image colors. The reason is that hue (H), saturation (S) and intensity (I) are magnitudes that can be interpreted intuitively and have a physical meaning [15]: the hue relates to the visual sensation of color purity, the saturation measures the proportion of white light diluted with pure color, and the intensity is related to the amount of light reflected by an illuminated object.

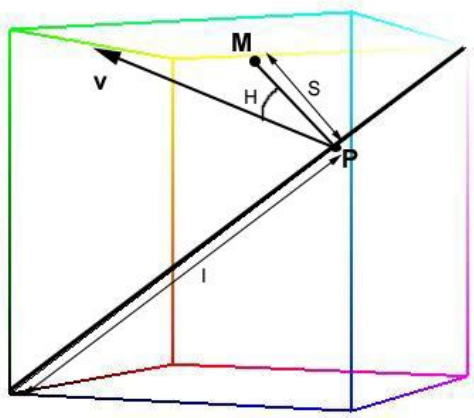


Figure 2: RGB color cube [14] and illustration of H, S and I magnitudes for a given color point $\boldsymbol{M} = (R, G, B)$.

In [3] the authors redefine the classical formulas for the conversion from RGB to HSI color space. They adapt the formulas to the following intuitive definitions of the three magnitudes: the intensity is the average of the 3 color components; the saturation measures the distance from the color point to the *gray axis* (locus of points in RGB space having identical color components); finally the hue is an angle around the gray axis[3]. Figure 2 illustrates the previous definitions and the conversion formulas are given by Equations (9)–(11).

$$I = \frac{R + G + B}{3}, \tag{9}$$

$$S = \sqrt{(R - I)^2 + (G - I)^2 + (B - I)^2}, \tag{10}$$

$$H = \text{sign}\left(-2(R - I) + (G - I) + (B - I)\right) \cdot \arccos\left(\frac{G - B}{\sqrt{2}S}\right). \tag{11}$$

With these definitions, the range of values of intensity, saturation and hue are, respectively, $[0, 255]$, $[0, 208.2066]$[4], $[-\pi, +\pi]$.

3.2 Achromatic Colors

From Equation (11) we can see that the hue magnitude cannot be defined for colors in the gray axis (i.e. colors with zero saturation). In practice, the hue component cannot reliably be computed for RGB colors whose saturation is small. For this reason, hue values are only computed for colors with large enough saturation, which are considered as *chromatic*, in contrast with low-saturated colors, which are called *achromatic*. In [7] the saturation threshold that discriminates chromatic from achromatic colors is computed using a simple argument: for a fixed intensity, at a distance S from the gray axis the maximum allowed number of color points is $2\pi S$, therefore, if we decide to quantize the hue component with Q different values S must be above $\frac{Q}{2\pi}$ to allow this quantization

[3]The angle is computed with respect to vector $\boldsymbol{v} = (0, 1/\sqrt{2}, -1/\sqrt{2})$, which is orthonormal to the gray axis (see Figure 2). The angle ranges from 0 to π, and a sign is assigned depending on the projection of vector $\overrightarrow{PM}$ onto $(-2, 1, 1)$, which is orthogonal to the gray axis and to the previous vector.

[4]The maximum corresponds to the saturation of pure red, green or blue colors, e.g. $(255, 0, 0)$.

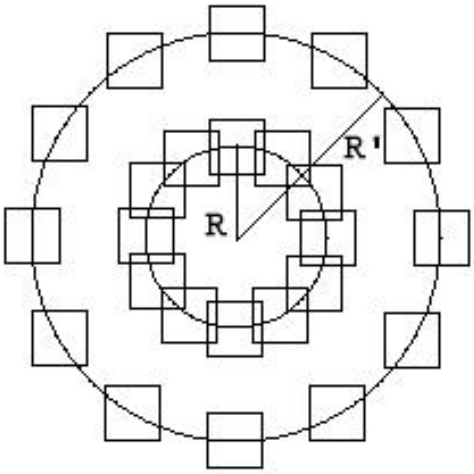


Figure 3: Illustration of the criterion used to fix the saturation threshold for the computation of hue values. If the saturation is too small (R), then the hue magnitude cannot be quantized into Q bins. This can be achieved only when the saturation is large enough (R').

(see Figure 3). The saturation threshold defines a cylinder in RGB space that contains the achromatic colors, the so-called *gray cylinder*.

3.3 Color Palette and Gray+Color Palette

In [7] and [3] it was proposed the following hierarchical algorithm for the construction of the color palette:

1. *Apply the FTC algorithm on the hue histogram of the image (computed from the colors not belonging to the gray-cylinder). Let S be the obtained segmentation.*

2. *Link each color of the gray cylinder to its corresponding interval $S_i = [s_i, s_{i+1}]$, according to its hue value. If the color belongs to the gray axis, then consider that its hue value is zero.*

3. *For each i, construct the saturation histogram of all the colors in the image whose hue belongs to S_i. Take into account the colors of the gray cylinder. Apply the FTC algorithm on the corresponding saturation histogram. For each i, let $\{S_{i,1}, S_{i,2}, \ldots\}$ be the obtained segmentation.*

4. *For each i and each j, compute and segment the intensity histogram of all the colors whose hue and saturation belong to S_i and $S_{i,j}$, including those in the gray cylinder.*

Remark that the above algorithm assumes that the input is a color image, and therefore cannot be applied to grayscale images. Moreover, as can be observed in Figures 4 to 6, it can produce palettes with an overrepresentation of dark colors. For these reasons, we propose in the current paper an alternative version of the algorithm in which Step 2 has been modified as follows:

2') Compute the intensity histogram for the pixels of the gray cylinder and apply the FTC algorithm to obtain a palette of gray values.

The rest of steps of the algorithm remain the same (except that all the references to the gray cylinder must now be removed). As a result of this modified algorithm both a gray and a color palette are obtained. In the next section, the images shown in Figures 4, 5 and 6 permit to compare the results of both versions of the algorithm.

A complete description of the automatic color palette algorithm (in its two versions) is provided in Algorithm 11. Several auxiliary functions are also described in Algorithms 5 to 9. In particular, the set of gray values that compose the gray palette is computed in Algorithm 6, the segmentation of the hue, saturation and intensity histograms is described in Algorithms 7 and 8. Finally, the hierarchical process for the computation of the color palette is described in Algorithm 9. The only parameters of the algorithm are the sizes of the bins (quantization factors) used to compute the hue, saturation and intensity histograms. The default values for these parameters are, respectively, 6°, 5 and 5, although they can be modified by the user.

Algorithm 5: Compute histogram

Input : array of values v, of size N
Input : number of histogram bins, L
Input : quantization of histogram bins, q
Output : histogram, $h = (h_i)_{i=1\cdots L}$

```
//Initialize all bins of histogram to zero
```
$h \longleftarrow 0$
for n *from* 1 *to* N **do**
  $i = \lfloor v[n]/q \rfloor + 1$ //index of bin, from 1 to L, $\lfloor\cdot\rfloor$ is floor operator
  $h_i \longleftarrow h_i + 1$ //increase bin count
return h

The lists of pixels associated to each mode of the histograms obtained with the hierarchical algorithm are recorded during the process. These lists are used to compute the average RGB color of all the pixels contributing to the same mode. These colors are displayed in a *color palette image*, which shows, in successive rows:

- the average intensity values of each mode of the histogram of intensities of the achromatic colors (that is, the gray palette), if the new version of the algorithm is used;
- the average RGB values of the pixels contributing to the modes of the hue histogram (first step of the algorithm);
- the average RGB values of the pixels contributing to the modes of the saturation histograms computed in the third step of the algorithm;
- the average RGB values of the pixels contributing to the modes of the intensity histograms computed in the last step of the algorithm (that is, the final color palette).

A second output of the algorithm is a segmentation of the input image using the colors in the palette. In [7] the colors in the final palette were used as seeds of a K-means algorithm in order to obtain a segmentation of the original image. In [3] a more direct approach was used: since the set of pixels associated to each mode of the hue, saturation and intensity histograms is known, the average RGB value of each set was computed and applied to all the pixels in the set. This is the approach that has been used in our implementation (see Algorithm 10).

The online demo that complements this paper lets the user upload any image, choose the quantization factors for the construction of the histograms, and displays both the color palette image and

Algorithm 6: Compute palette of grays

Input : color or grayscale image ($u = (H, S, I)$, or $u = I$)
Input : saturation threshold, $S_{\min}$
Input : number of bins in Intensity histogram, L
Input : quantization of histogram bins, q
Input : segmentation of the Intensity histogram into n meaningful modes, $s = \{s_0, \cdots, s_n\}$, with $1 = s_0 < s_1 < \cdots < s_n = L$
Output : gray level palette (list of main gray levels of u), $\mathcal{G} = \{G_1, \cdots, G_n\}$
Output : sets of pixels associated to each gray level in the palette, $P_{\mathcal{G}} = \{P_{G_1}, \cdots, P_{G_n}\}$

```
//Initialize output values
G ⟵ ∅
P_G ⟵ ∅
//Assign labels to histogram bins:  assign same label to all bins in range [s_{i-1}, s_i]
for each i ∈ {1, ··· , n} do
    for k = {s_{i-1}, ··· , s_i} do
        label[k] ⟵ i
//Compute average gray level and list of pixels associated to each segment [s_{i-1}, s_i]
//Initialize average values and lists of pixels
for each i ∈ {1, ··· , n} do
    G_i = 0
    P_{G_i} ⟵ ∅
for each pixel x of image u do
    //Consider gray value if the saturation of the pixel is below threshold S_min
    if S(x) ≤ S_min then
        k = ⌊I(x)/q⌋ + 1          //bin to which intensity value at pixel x contributes
        i = label[k]              //index of segment with intensity values in range [qs_{i-1}, qs_i]
        G_i = G_i + I(x)          //update total gray value associated to segment i
        P_{G_i} ⟵ P_{G_i} ∪ x     //add pixel to list associated to segment i
//Compute averages
for each i ∈ {1, ··· , n} do
    G_i = G_i/|P_{G_i}|           //|·| denotes the size of the set
    G ⟵ G ∪ G_i
    P_G ⟵ P_G ∪ P_{G_i}
return G, P_G
```

the segmentation result, for the two versions of the algorithm. It also permits to select the parameter ε of the FTC algorithm, which leads to finer or coarser segmentations of the histograms.

Algorithm 7: Hue segmentation

Input : color image $u = (H, S, I)$
Input : saturation threshold, $S_{\min}$
Input : number of bins in Hue histogram, L_H
Input : quantization of Hue histogram bins, q_H
Input : segmentation of the Hue histogram into $n+1$ meaningful modes, $s = \{s_0, \cdots, s_n\}$, with $s_0 < s_1 < \cdots < s_n$ (last mode is in $[s_n, s_0]$)
Input : `optionGray` boolean variable, if True compute gray+color palette, else only color palette
Output : list of pixels associated to each mode of the Hue histogram, $\mathcal{P}_\mathcal{H}$

`//Initialize output: empty list`
$\mathcal{P}_\mathcal{H} \longleftarrow \emptyset$
`//Assign labels to Hue histogram bins: assign same label to all bins in range` $[s_{i-1}, s_i]$
for *each* $i \in \{1, \cdots, n\}$ **do**
  for $k = \{s_{i-1}, \cdots, s_i\}$ **do**
    $\texttt{label}[k] \longleftarrow i$
`//Take into account that the Hue histogram is circular (last mode is in` $[s_n, s_0]$`)`
for $k = \{s_n, \cdots, L_H\}$ **do**
  $\texttt{label}[k] \longleftarrow n+1$
for $k = \{0, \cdots, s_0\}$ **do**
  $\texttt{label}[k] \longleftarrow n+1$
`//Compute list of pixels associated to each segment of Hue histogram`
for *each* $i \in \{1, \cdots, n+1\}$ **do**
  $P_{H_i} \longleftarrow \emptyset$
for *each pixel* $\boldsymbol{x}$ *of image* u **do**
  `//If optionGray is True consider only pixels whose saturation is above` $S_{\min}$`, else use all pixels`
  if `optionGray` *is* ***False or*** $S(\boldsymbol{x}) > S_{min}$ **then**
    $k = \lfloor H(\boldsymbol{x})/q_H \rfloor + 1$ `//bin to which hue value at pixel` $\boldsymbol{x}$ `contributes`
    $i = \texttt{label}[k]$ `//index of segment the bin belongs to`
    $P_{H_i} \longleftarrow P_{H_i} \cup \boldsymbol{x}$ `//add pixel to list associated to segment` i
for *each* $i \in \{1, \cdots, n+1\}$ **do**
  $\mathcal{P}_\mathcal{H} \longleftarrow \mathcal{P}_\mathcal{H} \cup P_{H_i}$
return $\mathcal{P}_\mathcal{H}$

Algorithm 8: Channel segmentation

Input : color image $u = (H, S, I)$
Input : channel identifier, $C \in \{S, I\}$, S (saturation) or I (intensity)
Input : list of image pixels P^0
Input : number of bins in channel histogram, L
Input : quantization of channel histogram bins, q
Input : segmentation of the channel histogram into n meaningful modes, $s = \{s_0, \cdots, s_n\}$, with $s_0 < s_1 < \cdots < s_n$
Output : list of pixels associated to each mode of the channel histogram, $\mathcal{P}$

```
//Initialize output:  empty list
```
$\mathcal{P} \longleftarrow \emptyset$
```
//Assign labels to channel histogram bins:  assign same label to all bins in range
```
$[s_{i-1}, s_i]$
for *each* $i \in \{1, \cdots, n\}$ **do**
  for $k = \{s_{i-1}, \cdots, s_i\}$ **do**
    label$[k] \longleftarrow i$

```
//Compute list of pixels associated to each segment of channel histogram
```
for *each* $i \in \{1, \cdots, n\}$ **do**
  $P_i \longleftarrow \emptyset$
for *each pixel* $\boldsymbol{x} \in P^0$ **do**
  $k = \lfloor C(\boldsymbol{x})/q \rfloor + 1$ `//bin to which channel value at pixel` $\boldsymbol{x}$ `contributes`
  $i =$ label$[k]$ `//index of segment the bin belongs to`
  $P_i \longleftarrow P_i \cup \boldsymbol{x}$ `//add pixel to list associated to segment` i
for *each* $i \in \{1, \cdots, n\}$ **do**
  $\mathcal{P} \longleftarrow \mathcal{P} \cup P_i$
return $\mathcal{P}$

Algorithm 9: Compute palette of colors

Input : color image u, RGB and HSI values are known for each pixel
Input : saturation threshold, $S_{\min}$
Input : number of bins of Hue, Saturation and Intensity histograms, L_H, L_S, L_I
Input : quantization of Hue, Saturation and Intensity histogram bins, q_H, q_S, q_I
Input : segmentation of the Hue histogram into $n+1$ meaningful modes, $s_H = \{s_0, \cdots, s_n\}$, with $s_0 < s_1 < \cdots < s_n$ (last mode is in $[s_n, s_0]$)
Input : parameter of FTC algorithm, ε
Input : `optionGray` boolean variable, if True compute gray+color palette, else only color palette
Output : color palette (list of main colors of u), $\mathcal{C} = \{\boldsymbol{C}_1, \cdots, \boldsymbol{C}_M\}$
Output : sets of pixels associated to each color in the palette, $P_{\mathcal{C}} = \{P_{C_1}, \cdots, P_{C_N}\}$

```
//Obtain list of pixels associated to each mode of the Hue histogram
P_H ⟵ Hue_segmentation(u, S_min, L_H, q_H, s_H, optionGray)          //Algorithm 7
//Compute Saturation histogram for each mode of Hue histogram, and segment it
for each P_{H_i} ∈ P_H do
    S' ⟵ ∅                                   //Empty array of saturation values
    for each x ∈ P_{H_i} do
        S' ⟵ S(x)                                          //Add value to array
        h_S ⟵ compute_histogram(S', L_S, q_S)                    //Algorithm 5
        //Segment Saturation histogram
        s_S ⟵ FTC_algorithm(h_S, ε)                              //Algorithm 1
        //Obtain list of pixels associated to each mode of the Saturation histogram
        P_{HS} ⟵ channel_segmentation(u, S, P_{H_i}, L_S, q_S, s_S)      //Algorithm 8
        //Compute Intensity histogram for each mode of Saturation histogram, and
        segment it
        for each P_{HS_i} ∈ P_{HS} do
            I' ⟵ ∅                              //Empty array of intensity values
            for each x ∈ P_{HS_i} do
                I' ⟵ I(x)                                  //Add value to array
                h_I ⟵ compute_histogram(I', L_I, q_I)                //Algorithm 5
                //Segment Intensity histogram
                s_I ⟵ FTC_algorithm(h_I, ε) //Algorithm 1
                //Obtain list of pixels associated to each mode of the Intensity
                histogram
                P_{HSI} ⟵ channel_segmentation(u, I, P_{HS_i}, L_I, q_I, s_I)   //Algorithm 8
                P_C ⟵ P_C ∪ P_{HSI}
//Compute average RGB color for each group of pixels
C ⟵ ∅
for each P_i ∈ P_C do
    C_i = (R_i, G_i, B_i) ⟵ (0, 0, 0)
    for each x ∈ P_i do
        (R_i, G_i, B_i) ⟵ (R_i, G_i, B_i) + (R(x), G(x), B(x))
    (R_i, G_i, B_i) ⟵ (R_i, G_i, B_i)/|P_i|          |·| denotes the number of pixels in the set
    C ⟵ C ∪ C_i
return C, P_C
```

Algorithm 10: Get segmented image

Input : color or gray scale image u
Input : gray level palette (list of main gray levels of u), $\mathcal{G} = \{G_1, \cdots, G_N\}$
Input : color palette (list of main colors of u), $\mathcal{C} = \{\boldsymbol{C}_1, \cdots, \boldsymbol{C}_M\}$
Input : sets of pixels associated to each gray level in the palette, $P_{\mathcal{G}} = \{P_{G_1}, \cdots, P_{G_N}\}$
Input : sets of pixels associated to each color in the palette, $P_{\mathcal{C}} = \{P_{C_1}, \cdots, P_{C_N}\}$
Output : segmented image v

```
for each set P_{G_i} ∈ P_G do
    for each pixel x ∈ P_{G_i} do
        v(x) = G_i                          //Assign constant gray level to all pixels in set
for each set P_{C_i} ∈ P_C do
    for each pixel x ∈ P_{C_i} do
        v(x) = C_i                          //Assign constant color to all pixels in set
return v
```

Algorithm 11: Automatic Color Palette

Input : color or grayscale image ($u = (R, G, B)$, or $u = I$), with values in range $[0, 255]$
Input : quantization parameters for the Hue, Saturation and Intensity histograms (q_H, q_S, q_I)
Input : parameter of FTC algorithm, ε
Input : `optionGray` boolean variable, if True compute gray+color palette, else only color palette
Output : color palette (list of main colors and/or gray levels in the image), $\mathcal{L}$
Output : segmentation of the input image using the colors in the palette, v

if u *is color image* **then**
  //Convert from RGB to HSI using Equations (9) to (11)
  //Use degrees to represent angles in the computation of Hue values (range [0, 360))
//Compute number of bins of histograms
$L_H = \lceil \frac{360}{qH} \rceil$ //number of bins in Hue histogram, $\lceil \cdot \rceil$ is ceil operator
$L_I = \lceil \frac{256}{qI} \rceil$ //number of bins in Intensity histogram, $\lceil \cdot \rceil$ is ceil operator
$L_S = \lceil \frac{208}{qS} \rceil$ //number of bins in Saturation histogram, $\lceil \cdot \rceil$ is ceil operator
//Compute minimum saturation value that prevents quantization problems in Hue histogram (see Section 3.2 for details)
$S_{\min} = L_H / 2\pi$
//Compute histogram of Hue values with enough saturation, and, if optionGray is True, histogram of Intensity values with low saturation
$H' \longleftarrow \emptyset$ //Empty array of valid Hue values
$I' \longleftarrow \emptyset$ //Empty array of low-saturated Intensity values
for *each pixel* $\boldsymbol{x}$ *of image* u **do**
  if $S(\boldsymbol{x}) > S_{min}$ **then**
    $H' \longleftarrow H(\boldsymbol{x})$ //Add value to array
  else
    if `optionGray` *is* ***True*** **then**
      $I' \longleftarrow I(\boldsymbol{x})$ //Add value to array
if `optionGray` *is* ***True*** **then**
  $h_I \longleftarrow$ `compute_histogram`(I', L_I, q_I) //Algorithm 5
  //Segment Intensity histogram
  $s_I \longleftarrow$ `FTC_algorithm`(h_I, ε) //Algorithm 1
  //Compute gray palette
  $\mathcal{G}, P_{\mathcal{G}} \longleftarrow$ `Compute_gray_palette`$(u, S_{\min}, L_I, q_I, s_I)$ //Algorithm 6
else
  $\mathcal{G} \longleftarrow \emptyset$, $P_{\mathcal{G}} \longleftarrow \emptyset$
$h_H \longleftarrow$ `compute_histogram`(H', L_H, q_H) //Algorithm 5
//Segment Hue histogram
$s_H \longleftarrow$ `FTC_algorithm_circular`(h_H, ε) //Algorithm 4
//Compute color palette
$\mathcal{C}, P_{\mathcal{C}} \longleftarrow$ `Compute_color_palette`$(u, S_{\min}, L_H, L_S, L_I, q_H, q_S, q_I, s_H, \varepsilon, \texttt{optionGray})$ //Algorithm 9
//Final color palette
$\mathcal{L} \longleftarrow \mathcal{G} \cup \mathcal{C}$
//Segmented output image
$v =\longleftarrow$ `get_segmented_image`$(u, \mathcal{G}, \mathcal{C}, P_{\mathcal{G}}, P_{\mathcal{C}})$ //Algorithm 10
return $\mathcal{L}$, v

4 Experimental Results

Figures 4, 5 and 6 show the results of applying both versions of the algorithm to different images, using the default values of the parameters. For each version of the algorithm, the following images are displayed:

1) H-segmented image, where the same color has been assigned to all the pixels contributing to the same mode of the hue histogram. The assigned color is the average RGB value of the set of pixels. The colors displayed in this image are the ones shown in the first row of the color palette. If the new version of the algorithm is used, the color of the achromatic pixels contributing to the same mode of the intensity histogram is also replaced by the average intensity value of the set of pixels (the values in the gray palette).

2) HS-segmented image, where the same color has been assigned to all the pixels contributing to the same mode of the saturation histograms computed in the third step of the algorithm. The assigned color is the average RGB value of the set of pixels. The colors displayed in this image are the ones shown in the second row of the color palette.

3) HSI-segmented image, the final segmentation results, whose colors are the ones displayed in the last row of the color palette image.

4) Color palette image.

We observe that with the new version of the algorithm we obtain a more compact representation of the image colors: the palette contains less colors and a fewer number of similar low-saturated colors is obtained in the final palette (observe the last row in the color palette images of the three figures). On the other hand, the final segmentations obtained with both versions of the algorithm are very similar. For these reasons, in the following experiments, only the new version of the algorithm (the so-called *gray+color* version) shall be used.

Figures 7, 8 and 9 explore the ability of the algorithm to preserve rare colors in the images, that is, colors corresponding to a small fraction of the image pixels. In Figure 7 we observe that the red color corresponding to the flowers has effectively been preserved. However, this is not the case with the yellow color of the rope in Figure 8 or the red color of the flowers in Figure 9. By observing the first row of the color palettes and the H-segmentations obtained for these images we see that, in the first step of the algorithm, these rare colors have been grouped together with other colors with similar hues, and the subsequent steps of the hierarchical method are unable to discriminate them based on their saturation or intensity.

In order to improve the segmentation of the hue values in the first step of the algorithm, a finer quantization can be used. The default quantization factors for hue, saturation and intensity (6°, 5, and 5) provide good results in most cases, as shown in Figures 4 to 7, although in some cases the results can be improved using different factors. Figures 10 to 13 display the segmentation results obtained for different sets of quantization factors.

In Figures 10 and 11 decreasing quantization factors for the hue have been used. We observe that the rare colors are correctly represented for finer quantizations. However, a side effect of using a finer hue quantization is that the threshold that separates achromatic from chromatic colors increases (see Section 3.2), and therefore a higher number of colors are considered achromatic and thus represented by gray values.

In Figures 12 and 13 different quantizations for the saturation and intensity components have been tested, for a fixed quantization of the hue. We observe that a finer quantization of the saturation permits to distinguish more image details (see the mountains on the left side of the image in Figure 12, or the letters of the boat in Figure 13). As the quantization factor increases less details are visible,

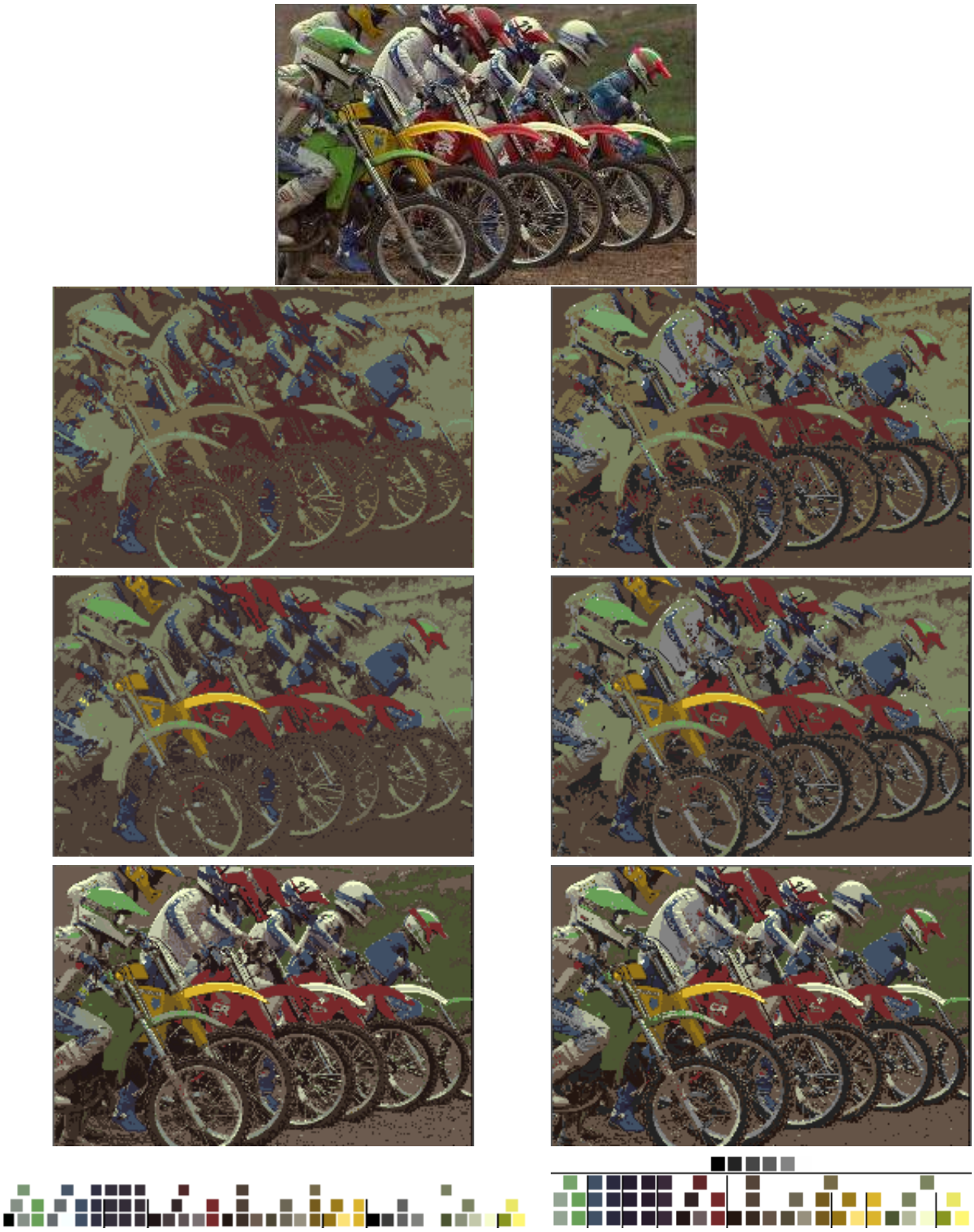


Figure 4: Original image and results of applying the original version (left) and the new version (right) of the Automatic Color Palette algorithm. From top to bottom: H-segmented, HS-segmented, HSI-segmented images and color palette image. The palette on the left side contains 36 colors (last row of the palette image), while the one on the right 6+24 (6 for the gray palette and 24 for the color palette).

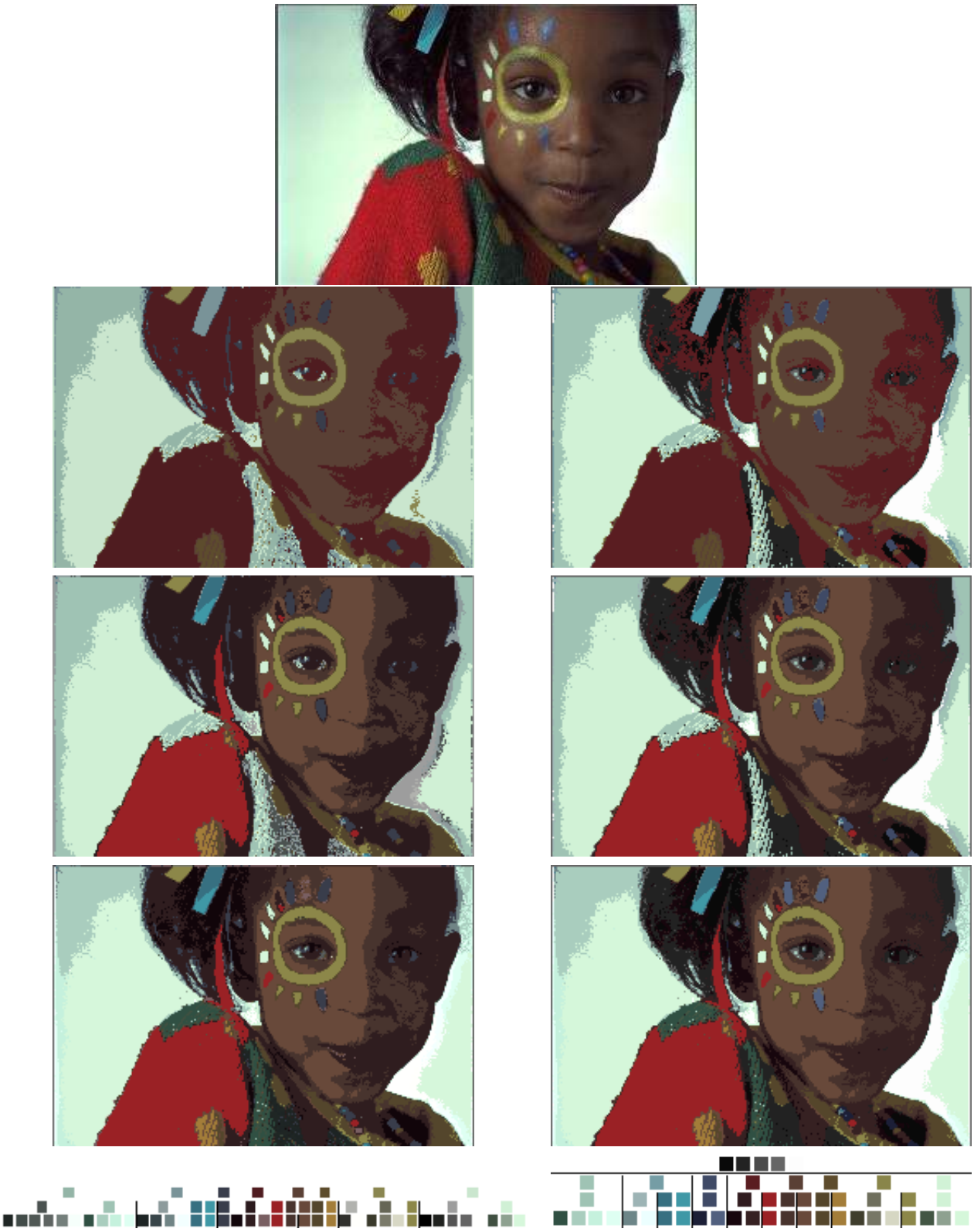


Figure 5: Original image and results of applying the original version (left) and the new version (right) of the Automatic Color Palette algorithm. From top to bottom: H-segmented, HS-segmented, HSI-segmented images and color palette image. The palette on the left side contains 39 colors (last row of the palette image), while the one on the right 5+24 (5 for the gray palette and 24 for the color palette).

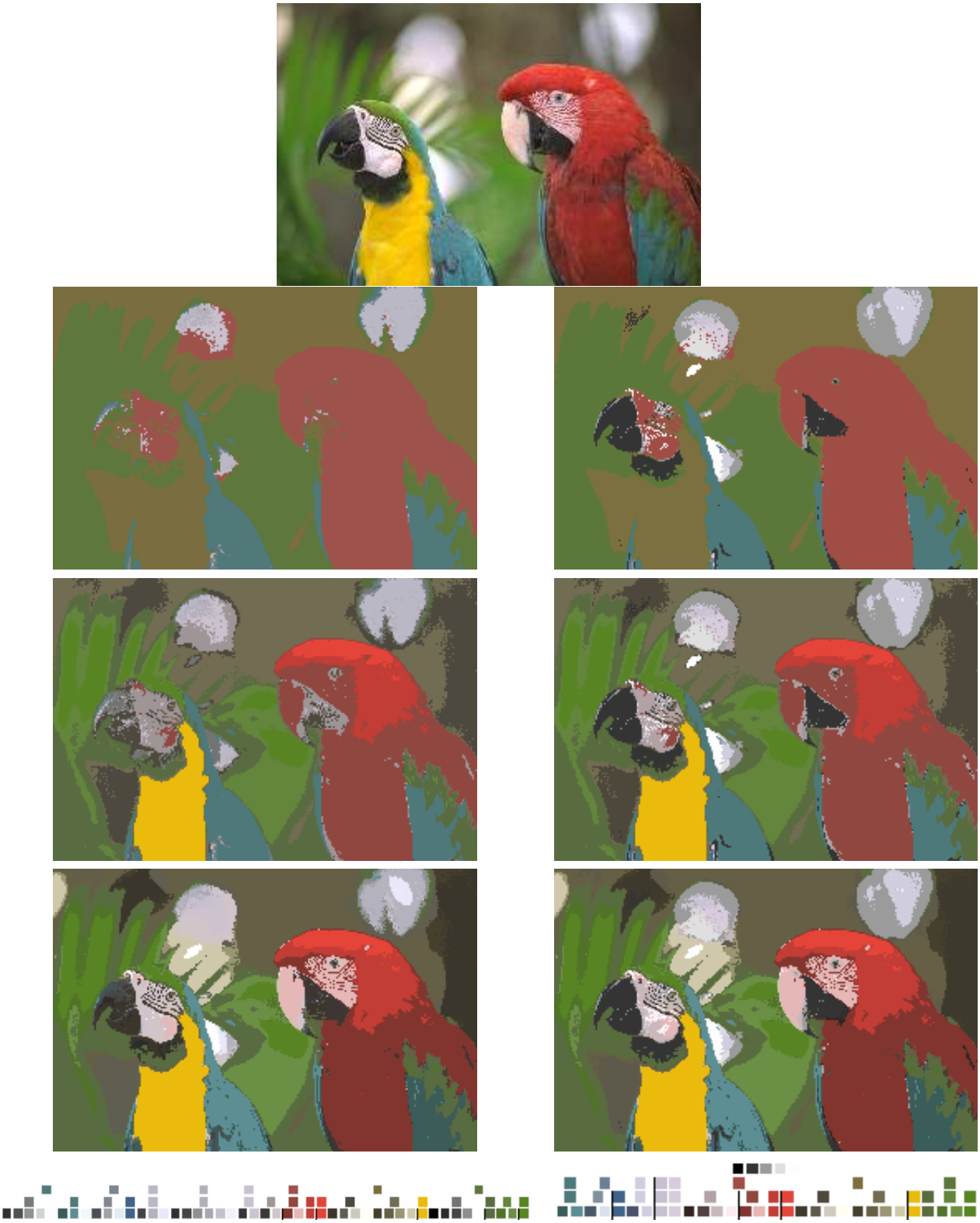


Figure 6: Original image and results of applying the original version (left) and the new version (right) of the Automatic Color Palette algorithm. From top to bottom: H-segmented, HS-segmented, HSI-segmented images and color palette image. The palette on the left side contains 47 colors (last row of the palette image), while the one on the right 5+30 (5 for the gray palette and 30 for the color palette).

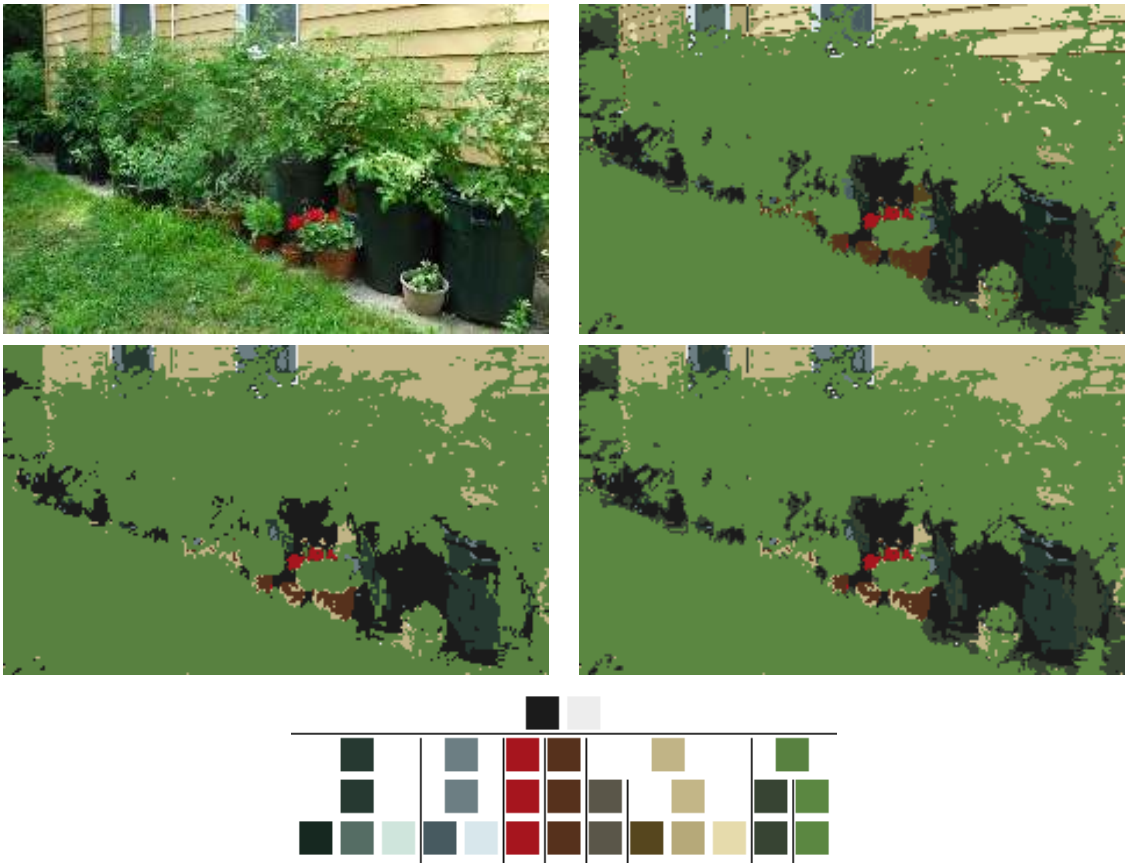


Figure 7: From left to right and from top to bottom: original image, final segmentation, H-segmentation, HS-segmentation and gray+color palette. The gray palette is composed of 2 gray levels and the color palette of 13 colors.

although little differences can be observed in these images between the results for quantization factors 5 and 10. Similar remarks can be made concerning the quantization factor of the intensity component: for finer quantizations more details are visible in some parts of the images (see the foam in the river, in the bottom-left image of Figure 13), while less colors are obtained as the quantization factor increases. As expected, the number of colors in the palette increases when using finer quantizations. Variations in the saturation quantization affect only to the color palette, but not to the gray palette, while the intensity quantization affects both palettes.

Finally, the effect of the parameter ε of the FTC algorithm on the palette results is tested. For the images in Figures 8 and 9, for which the rare colors were not detected using the default parameters, we show the results obtained with increasing values of ε, (the quantization parameters are fixed to their default values). We observe that the number of detected colors increases, since the histogram segmentations are finer. Eventually, the rare colors are included in the palette. Conversely, if we want to obtain a palette with less colors, we should decrease the value of the parameter.

Figure 8: From left to right and from top to bottom: original image, final segmentation, H-segmentation, HS-segmentation and gray+color palette. The gray palette is composed of 3 gray levels and the color palette of 19 colors.

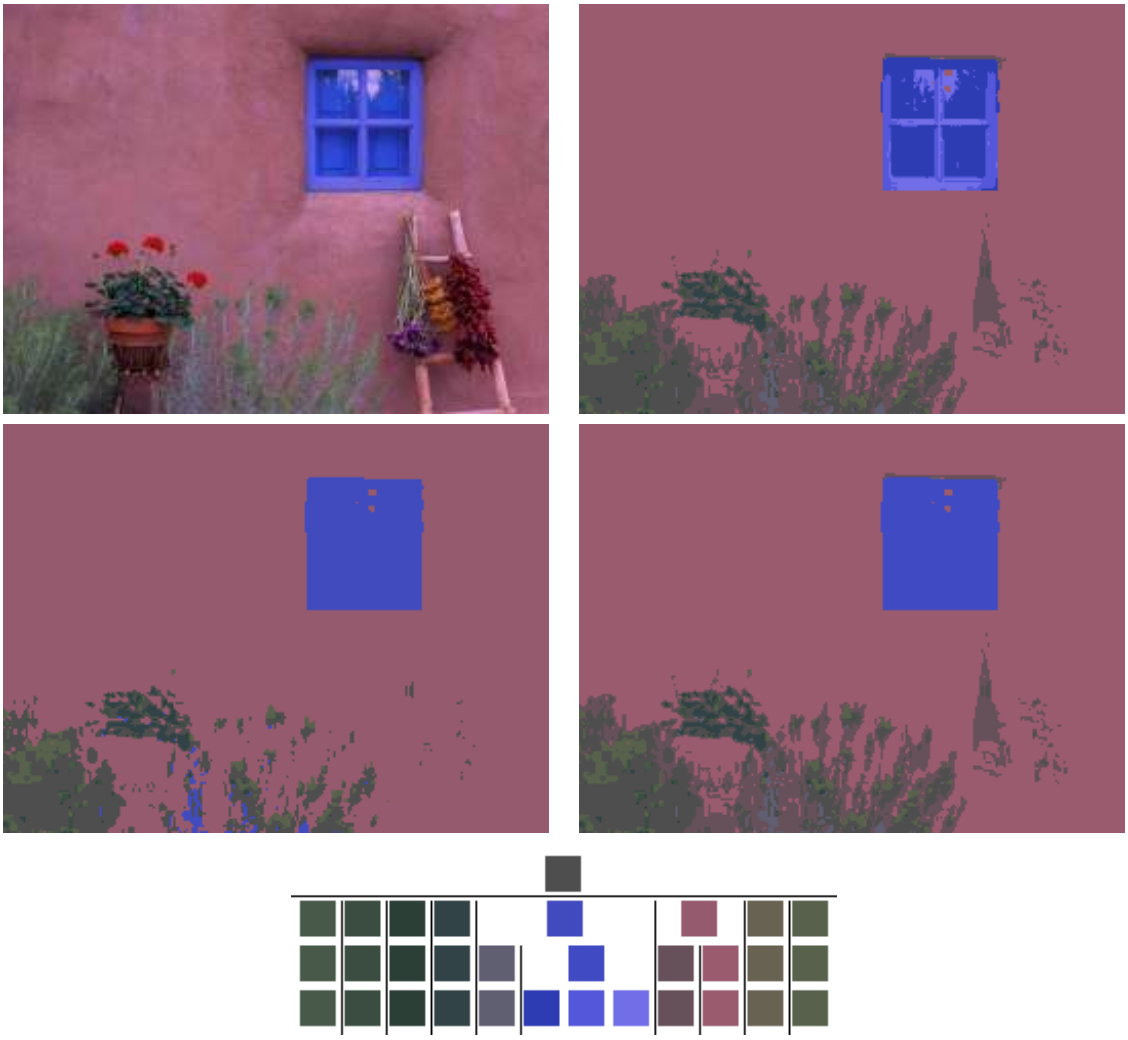


Figure 9: From left to right and from top to bottom: original image, final segmentation, H-segmentation, HS-segmentation and gray+color palette. The gray palette is composed of 1 gray level and the color palette of 12 colors.

$qH = 3$, $qS = 5$, $qI = 5$ $qH = 2$, $qS = 5$, $qI = 5$ $qH = 1$, $qS = 5$, $qI = 5$

Figure 10: Effect of the variation of the quantization of the hue values.

$qH = 3,\ qS = 5,\ qI = 5$ $qH = 2,\ qS = 5,\ qI = 5$ $qH = 1,\ qS = 5,\ qI = 5$

Figure 11: Effect of the variation of the quantization of the hue values.

Original $qH = 6,\ qS = 5,\ qI = 5$ (5+23)

$qH = 6,\ qS = 2,\ qI = 5$ (5+33) $qH = 6,\ qS = 10,\ qI = 5$ (5+23)

$qH = 6,\ qS = 5,\ qI = 2$ (9+36) $qH = 6,\ qS = 5,\ qI = 10$ (3+21)

Figure 12: Effect of the variation of the quantization of the saturation and intensity values. The values in parentheses are the number of colors in the gray and color palettes.

Original

$qH = 6,\ qS = 5,\ qI = 5$ (5+20)

$qH = 6,\ qS = 2,\ qI = 5$ (5+35)

$qH = 6,\ qS = 10,\ qI = 5$ (5+19)

$qH = 6,\ qS = 5,\ qI = 2$ (11+49)

$qH = 6,\ qS = 5,\ qI = 10$ (3+19)

Figure 13: Effect of the variation of the quantization of the saturation and intensity values. The values in parentheses are the number of colors in the gray and color palettes.

$\varepsilon = 1$ (default) (3+19)

$\varepsilon = 10$ (3+24)

$\varepsilon = 100$ (8+44)

Figure 14: Effect of the variation of the parameter ε of the FTC algorithm. The values in parentheses are the number of colors in the gray and color palettes.

$\varepsilon = 1$ (default) (1+12)

$\varepsilon = 10$ (1+25)

$\varepsilon = 100$ (1+64)

Figure 15: Effect of the variation of the parameter ε of the FTC algorithm. The values in parentheses are the number of colors in the gray and color palettes.

5 Conclusions

We have presented in this paper a detailed description of an algorithm for the automatic parsing of a 1D histogram into its meaningful modes, that is, intervals where the histogram follows a unimodal distribution.

This algorithm has then been used to analyze, in a hierarchical way, the histograms of the hue, saturation and intensity components of a digital image, in order to compute a minimum representation of the image colors, the so-called color palette.

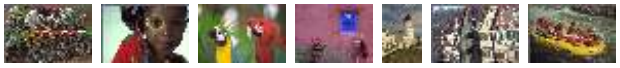
A slight modification of the original color palette estimation method (published in [7]) has been proposed, involving the treatment of the achromatic colors of the image, which leads to a gray+color palette which, in general, provides a more compact representation than the original method.

Several experiments show that the presented method is able, in general, to obtain a minimum and accurate description of the image colors, although rare colors are sometimes missed. Further improvements of the method (such as the combination of the palettes obtained with different quantization factors) could permit the preservation of these colors in the final result.

Acknowledgment

The authors acknowledge the Ministerio de Ciencia, Innovación y Universidades (MCIU), the Agencia Estatal de Investigación (AEI) and the European Regional Development Funds (ERDF) for its support to the project TIN2017-85572-P. This work has also been sponsored by the Comunitat Autonoma de les Illes Balears through the Direcció General de Política Universitària i Recerca with funds from the Tourist Stay Tax Law ITS 2017-006 (PRD2018/26).

Image Credits

 Kodak image dataset[5]

 Wikimedia commons CC0[6]

References

[1] M. Alvioli, A. C. Mondini, F. Fiorucci, M. Cardinali, and I. Marchesini, *Topography-driven satellite imagery analysis for landslide mapping*, Geomatics, Natural Hazards and Risk, 9 (2018), pp. 544–567. https://doi.org/10.1080/19475705.2018.1458050.

[2] M. Ayer, H. Brunk, G. Ewing, W. Reid, and E. Silverman, *An empirical distribution function for sampling with incomplete information*, Annals of Mathematical Statistics, 26 (1955), pp. 641–647.

[3] Ana Belén Petro Balaguer, *Analytical methods for the study of color in digital images*, PhD thesis, Universitat de les Illes Balears, 2006.

[4] L. Birgé, *The Grenander estimator: A nonasymptotic approach*, Annals of Mathematical Statistics, 17 (1989), pp. 1532–1549. https://doi.org/10.1214/aos/1176347380.

[5] http://www.cs.albany.edu/~xypan/research/snr/Kodak.html

[6] https://commons.wikimedia.org/wiki/File:Tomato_plants_growing_July_2013_in_garbage_cans.JPG

[5] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons Inc., 1991. ISBN 978-0-471-24195-9.

[6] J. Delon, A. Desolneux, J.L. Lisani, and A.B. Petro, *Histogram analysis and its applications to fast camera stabilization*, in 11th International Conference on Signals and Electronic Systems (IWSSIP), 2004.

[7] ——, *Automatic color palette*, Inverse Problems & Imaging, 1 (2007), pp. 265–287. https://doi.org/10.3934/ipi.2007.1.265.

[8] ——, *A nonparametric approach for histogram segmentation*, IEEE Transactions on Image Processing, 16 (2007), pp. 253–261. https://doi.org/10.1109/TIP.2006.884951.

[9] A. Desolneux, L. Moisan, and J-M. Morel, *Meaningful alignments*, International Journal of Computer Vision, 40 (2000), pp. 7–23. https://doi.org/10.1023/A:1026593302236.

[10] ——, *A grouping principle and four applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 508–513. https://doi.org/10.1109/TPAMI.2003.1190576.

[11] A. Desolneux, L. Moisan, and J-M. Morel, *From Gestalt theory to image analysis: a probabilistic approach*, Interdisciplinary Applied Mathematics, Springer, 2008. ISBN 978-0-387-74378-3.

[12] U. Grenander, *Abstract Inference*, Wiley, 1981.

[13] J.L. Lisani, L. Rudin, P. Monasse, J-M. Morel, and P. Yu, *Meaningful automatic video demultiplexing with unknown number of cameras, contrast changes, and motion*, in IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.

[14] J.L. Lisani, T.Buades, and J-M. Morel, *Image Color Cube Dimensional Filtering and Visualization*, Image Processing On Line, 1 (2011), pp. 57–69. https://doi.org/10.5201/ipol.2011.blm-cdf.

[15] G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, Wiley, 1984, ch. 3. ISBN 978-0-471-39918-6.