Chapter Number

Protecting the color information by hiding it

Marc CHAUMONT and William PUECH
University of Nîmes, University of Montpellier 2, CNRS UMR 5506- LIRMM, France

1. Introduction

Few work have been proposed for image color protection. A color protection algorithm should give some comprehensive information about the image (such as its grey-level version) but **should hide securely the color information**. The proposed algorithms are thus specific but take into account recent signal processing concepts such as quantization, compression and data-hiding. Figure 1 illustrates a possible use of the color protection where only secret key owners may rebuild the color image.

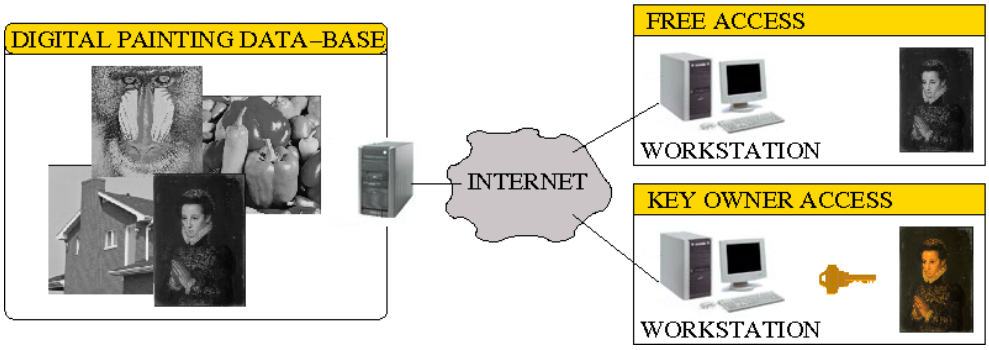


Fig. 1. Illustration of possible color protection use.

This chapter presents some previously published solutions and proposes new solutions built on data-hiding methods. Remember that many methods propose solutions to **hide information** by using the decomposition of a color image in an *index* image and a color palette (Fridrich, 1998, Wu et al., 2003, Tzeng et al., 2004). The data-hiding may occur in the *index* image (Fridrich, 1998) or in the color palette (Wu et al., 2003, Tzeng et al., 2004). Nevertheless, none of those techniques tries to protect the color information by hiding the color palette in the *index* image. Other approaches such that (Campisi et al., 2002, Queiroz & Braun, 2006) based on wavelet decomposition and sub-band substitution propose solutions to embed the color information in a grey-level image. Their areas are image printing for (Queiroz & Braun, 2006), and perceptive compression for (Campisi et al., 2002). Even if those techniques embed the color information, their approaches and their purposes are clearly different since they do not try to preserve the best color quality while ensuring security.

In previous work (Chaumont & Puech, 2007a, Chaumont & Puech, 2007b), we presented two new approaches in order to protect the color information by hiding the color palette in the *index* image. In (Chaumont & Puech, 2007a) we propose to sort the colors of the color palette in order to get an *index* image which is near of the luminance of the original color image. In the same time, we get a color palette whose consecutive colors are close. In (Chaumont & Puech, 2007b) the approach is completely different and relies on a function optimization of the global problem formulation. In those two previous work, the security and the color quality requirements were not fully addressed. In this chapter, we give new directions of analysis and improve the security and the quantized color image quality.

This chapter will first remind two solutions quite close from the image color protection problem: the image printing algorithm (Queiroz & Braun, 2006) and the perceptive compression algorithm (Campisi et al., 2002). We will then remind the two specific solutions: the fuzzy approach (Chaumont & Puech, 2007b) and the re-ordering approach (Chaumont & Puech, 2007a). Finally, we will present a better solution for this problem which consists to hide a 512 color palette in an 8-bit-grey-level image.

2. Two substitution-based approaches in the wavelet domain

2.1. Queiroz & Braun color hiding

The solution proposed by (Queiroz & Braun, 2006) is specific to the image printing problem. When a document owning color images has to be distributed in a society, it is still often printed onto black-and-white printers. Graphics, pie charts, or other color images may then become unreadable. The proposed solution is to transform a color image in a readable *grey-level* image even in presence of colors of same luminance. What is interesting us in the (Queiroz & Braun, 2006) solution is not the readability aspect but the reversibility of their transformation from grey-level to color. Indeed, it is possible to recover a visually close approximation of the original color image from the *grey-level* image.

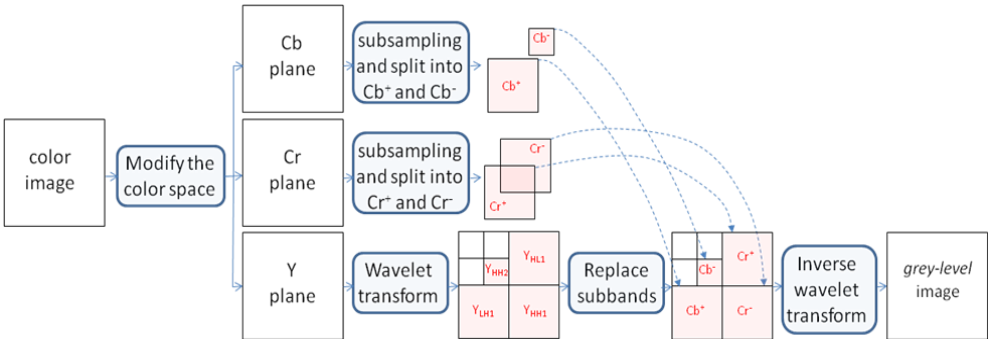


Fig. 2. Sub-band replacement for color embedding of (Queiroz & Braun, 2006).

The technique, illustrated in Figure 2, is very simple and consists:

a) to express the color image into the Y, Cr, Cb color space,
b) to subsample by four the two planes Cr and Cb,

c) to derive two planes Cr^+ and Cr^- (resp. Cb^+ and Cb^-) from the subsampled Cr (resp. Cb) plane; The plane Cr^+ (resp. Cb^+) is a copy of the subsampled Cr (resp. Cb) plane where negative values are set to 0 and the plane Cr^- (resp. Cb^-) is a copy of the subsample Cr (resp. Cb) plane where positive values are set to 0,
d) to decompose the Y plane with a 2-levels discrete wavelet transform,
e) to substitute the LH1 by Cb^+, HL1 by Cr^+, HH1 by Cr^- and HH2 by a ¼ subsampling of Cb^-,
f) to inverse the wavelet transform in order to produce a *grey-level* image embedding its color information.

To recover the embedded color information i.e the Y, Cr, Cb planes, from the *grey-level* image and then rebuild a color image, one should:

a) apply a 2-levels wavelet transform,
b) extract Cb^+, Cr^+, Cr^-, and Cb^-.
c) up-sample the Cb^- plane,
d) retrieve Cr and Cb planes such that $Cr = |Cr^+|-|Cr^-|$ and $Cb = |Cb^+|-|Cb^-|$, and up-sample them,
e) set sub-bands coefficients of HL1, LH1, HH1 and HH2 to 0 and apply the invert wavelet transform.

Figure 3 illustrates the recovering principle.

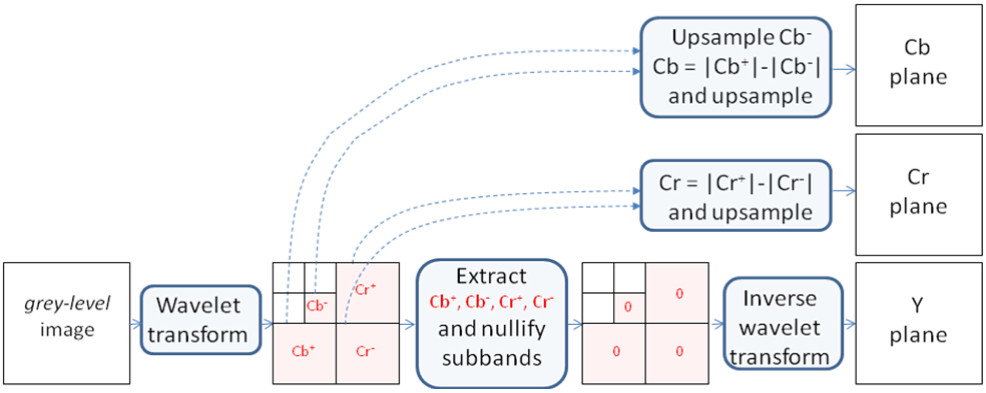
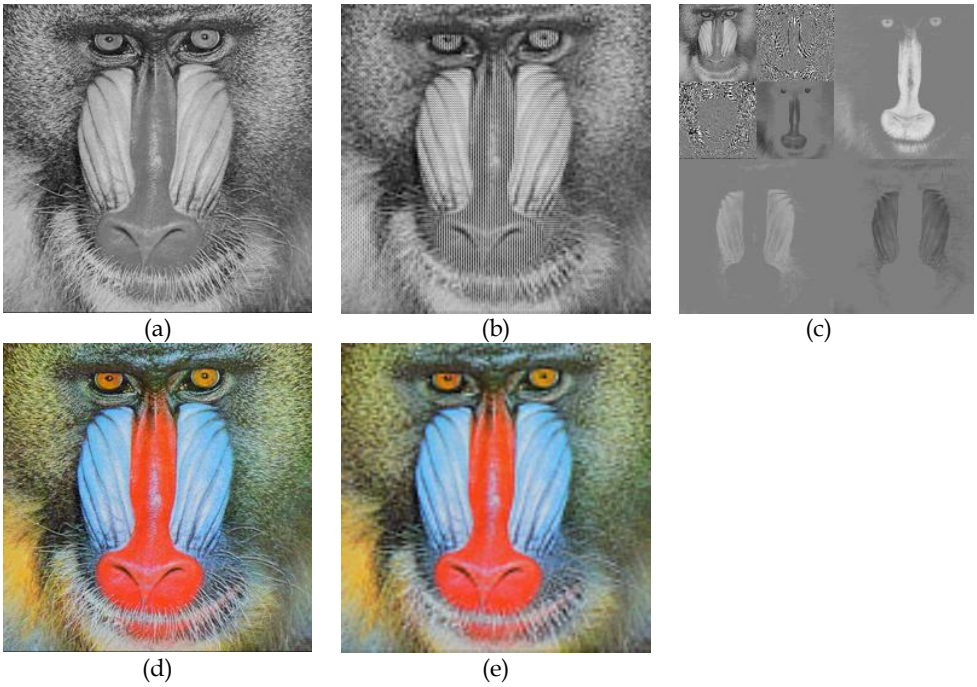


Fig. 3. Recovering of a color image from a *grey-level* image (Queiroz & Braun, 2006).

This substitution approach in the wavelet domain is interesting for its potential for hiding the color information into a grey-level image. Figure 4 shows some key images of the process for the baboon image. One could nevertheless remark that the retrieved color image (Figure 4.e) from the *grey-level* image has a poor visual quality. We could note some blurring and strong ringing effects. Those artefacts are due to the poor quality of the retrieve Y plane since lots of wavelet sub-bands are nullify. In Table 1 we should essentially remark that PSNRs between the original color image and the rebuild color image are below 31 dB. In conclusion, this approach is only useful for printing applications. We will see in the next section another similar approach (Campisi et al., 2002) which gives higher PSNR and images visually more pleasant.



(a) (b) (c)

(d) (e)

Fig. 4. Application of the *substitution-based approach* of (Queiroz & Braun 2006): a) Luminance of the original color image, b) *Grey-level* image embedding the chrominance planes, c) Wavelet decomposition of the *grey-level* image, d) Original color image, e) Rebuilt color image from the *grey-level* image.

Images	PSNR *(luminance, grey-level)*	PSNR *(original color, rebuilt)*
baboon	21.03 dB	23.93 dB
barbara	23.93 dB	26.33 dB
airplane	26.47 dB	28.56 dB
peppers	21.24 dB	28.82 dB
lena	21.02 dB	30.31 dB
house	25.18 dB	30.75 dB

Table 1. PSNR between the luminance image and the *grey-level* image and PSNR between the original color image (RGB space) and the rebuilt color image (RGB space) using the *grey-level* image embedding its color information.

2.2. Campisi et al. color hiding

The objective of the solution proposed by (Campisi et al., 2002) is to improve the compression efficiency of a color image. The idea is to transform a color image into a single grey-level image which embeds the chrominance planes. The solution principle is thus very similar to (Queiroz & Braun 2006) approach. Nevertheless, the distortion between the original color image and the rebuilt one is smaller.

The approach, illustrated in Figure 5, consists in:

a) expressing the color image into the Y, I, Q color space,
b) sub-sampling by 16 the two planes I and Q in order to obtain I_{LL2} and Q_{LL2} planes; Note that this sub-sampling is process by applying a 2-levels wavelet Daubechies 9/7 (Daubechies & Sweldens, 1998) decomposition and keeping the low-pass sub-band of I and Q,
c) applying a 1-level wavelet Daubechies decomposition to the Y plane and re-apply a 1-level wavelet decomposition to the sub-bands HL and LH; The decomposition of sub-band HL (resp. LH) give four sub-bands and the low-pass one is noted $Y_{ll,HL}$ (resp. $Y_{ll,LH}$),
d) normalizing the two planes I_{LL2} and Q_{LL2} with values N_I and N_Q; those values will be transmitted as side information,
e) substitute the sub-band $Y_{ll,HL}$ (resp. $Y_{ll,LH}$) by I_{LL2} (resp. Q_{LL2}),
f) compress the data.

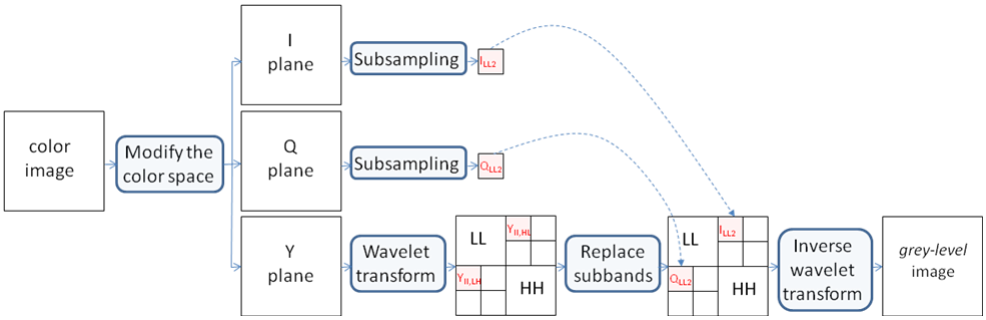


Fig. 5. Sub-band replacement for color embedding of (Campisi et al., 2002).

To recover the embedded color information i.e Y, I, Q planes, from the *grey-level* image and then rebuild a color image, one should:

a) apply the wavelet transforms,
b) extract I_{LL2} and Q_{LL2} and rescale them thanks to normalizing values N_I and N_Q,
c) up-sample the I_{LL2} and Q_{LL2} in order to retrieve I and Q planes,
g) set sub-bands coefficients of $Y_{ll,HL}$ and $Y_{ll,LH}$ to 0 and apply the invert wavelet transforms.

Figure 6 illustrates the recovering principle.

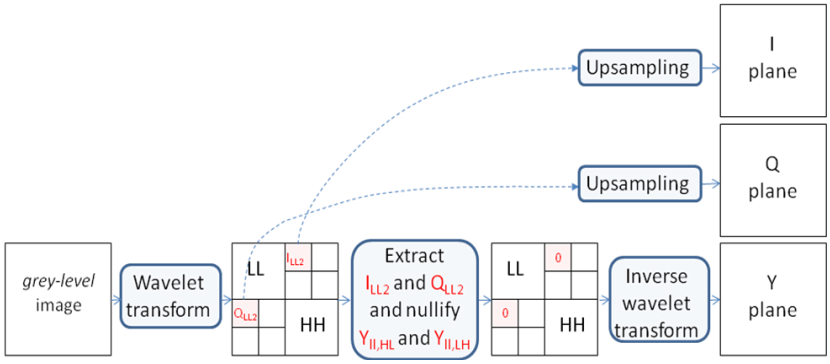
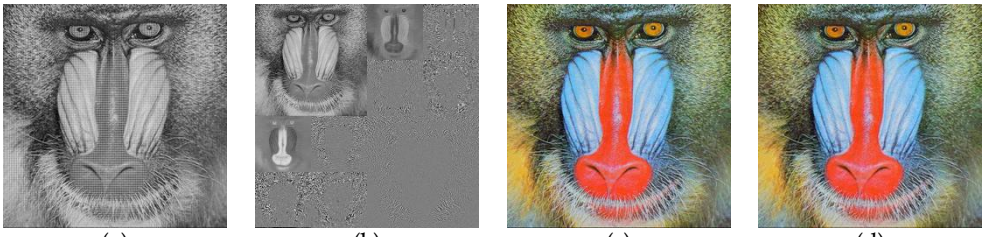


Fig. 6. Recovering of a color image from a *grey-level* image (Campisi et al., 2002).

Figure 7 shows some key images of the process for the baboon image. One could remark that the reconstructed color image (Figure 7.d) from the *grey-level* image has a good visual quality. Since, the Campisi et al. approach does not degrade too much the Y component and that chrominance planes are good enough, the rebuilt color image is visually pleasant. We just notice some small ringing effects. In Table 2 we should essentially remark that the PSNRs between the original color image and the rebuild color image are above 29 dB. Thus, the Campisi et al. approach performs better objective and subjective results than (Queiroz & Braun 2006). We will see in the next section, that the palette based approach give rebuilt color images of even better quality.



(a) (b) (c) (d)

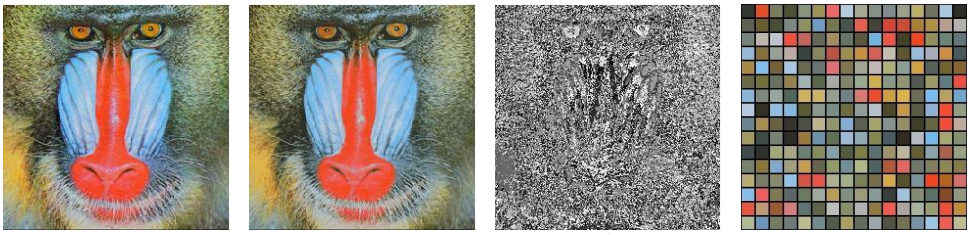
Fig. 7. Application of the *substitution-based approach* of (Campisi et al., 2002): a) *Grey-level* image embedding the chrominance planes, b) Wavelet decomposition of the *grey-level* image, c) Original color image, d) Rebuilt color image from the *grey-level* image.

Images	PSNR *(luminance, grey-level)*	PSNR *(original color, rebuilt)*
baboon	27.37 dB	29.8 dB
barbara	30.61 dB	31.75 dB
peppers	25.82 dB	32.36 dB
airplane	34.12 dB	32.58 dB
house	30.76 dB	31.76 dB
lena	26.91 dB	36.75 dB

Table 2. PSNR between the luminance image and the *grey-level* image and PSNR between the original color image (RGB space) and the rebuilt color image (RGB space) using the *grey-level* image embedding its color information.

3. Two palette-based approaches

In order to obtain a grey-level image embedding its color information, we decompose a color image in an *index* image and a color palette. Figure 8 illustrates a decomposition of the Baboon color image into an *index* image and a color palette. Note that the decomposition is a quantization since the total number of colors is reduced. The color palette is then hidden in the *index* image. The *index* image should be similar to the luminance of the color image, the embedding process should be of week magnitude and the color palette should be cleverly ordered. We proposed in previous work two solutions for this problem: **the fuzzy approach** (Chaumont & Puech, 2007b) and the **reordering approach** (Chaumont & Puech, 2007a).



Original Quantized *Index* Color palette

Fig. 8. Decomposition of a color image into an *Index* image and a color palette.

3.1. The fuzzy approach

The originality of the fuzzy approach is to propose a solution for the very constrained decomposition problem. Thus, the main contribution of this work is the proposition of an energetic function in order to model the decomposition of the color image. The optimization of the energetic function leads to the obtaining of a well suit *index* image and a well sorted color palette. The fuzzy approach is then decomposed in two parts: the energetic function optimisation and the data-hiding.

3.1.1. The problem formulation and its resolution

The goal of the first step is to find an *index* image and a color palette with the following constraints:

- the *index* image should be close from the luminance of the original color image,
- the color quantized image should be close from the color image,
- and, for data-hiding purpose, the color palette should own consecutive couples of close color.

The problem of the computation of a color palette made of couples of close colors and an *index* image similar to the luminance may be expressed by three constraints. Mathematically this comes to found the K colors $C(k)$ (C is the color palette) and the $P_{i,k}$ ownership values giving the degree of belongingness of a pixel i to the k^{th} color. Note that $P_{i,k}$ belongs to $[0,1]$ and are named fuzzy membership values in fuzzy c-mean clustering approach (Dunn, 1974). Also note that the $P_{i,k}$ give indirectly the *index* image such that: $Index(i) = \arg_k \max_k P_{i,k}$.

Thus, we are looking to minimize the above energetic model in order to obtain

$\forall i \in [1,N], \forall k \in [1,K]$, $P_{i,k}$ and *C(k)*:

$$E = \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{K} P_{i,k}^{m}(C(k)-I(i))^2}_{first\ term} + \lambda_1 \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{K} P_{i,k}^{m}(Y(i)-k)^2}_{second\ term} + \lambda_2 \underbrace{\sum_{k|k\in[1..K]\ and\ k\ is\ odd} (C(k)-C(k+1))^2}_{third\ term}, \quad (1)$$

with *I* the color image, *Y* the luminance image, λ_1 and λ_2 two scalar values and $m \in]1, \infty[$ the fuzzy coefficient tuning the equi-probability. Note that *m* is set to 2 for computational complexity reduction.

The first term is expressing the constraint of color quantization. The aim is to found the best representative *K* colors. The second term stands for getting the *index* image the nearest to the luminance image *Y*. The last term constrain couples of consecutive color from the palette to be close.

The minimization of Equation (1) such that:

$$\{P_{i,k}, C(k)\} = \arg\min_{\{P_{i,k}, C(k)\}} E,$$

is performed iteratively in a two steps loop as in conventional fuzzy c-mean algorithms. In the first step, colors *C(k)* are updated, given $P_{i,k}$, by solving the linear system below:

$$\underline{\forall k\ odd:}$$
$$(\lambda_2 + \sum_{i=1}^{N} P_{i,k}^{m}) \times C(k) - \lambda_2 \times C(k+1) = \sum_{i=1}^{N} P_{i,k}^{m} I(i),$$

$$\underline{\forall k\ even:}$$
$$-\lambda_2 \times C(k-1) + (\lambda_2 + \sum_{i=1}^{N} P_{i,k}^{m}) \times C(k) = \sum_{i=1}^{N} P_{i,k}^{m} I(i).$$

In the second step, $P_{i,k}$ (with *m*=2) are updated given the colors *C(k)* with:

$$P_{i,k} = \frac{(\sum_{l=1}^{l=K} \frac{1}{2\times((C(l)-I(i))^2+\lambda_1(Y(i)-l)^2)})^{-1}}{2 \times ((C(k)-I(i))^2 + \lambda_1(Y(i)-k)^2)}$$

Mathematical details are given in (Chaumont & Puech, 2007b).

3.1.2. Spatial data hiding

For this approach, we have used an algorithm to embed the color palette information in the Least Significant Bit (LSB) of an image of *N* pixels. The objective is thus to embed a message *M* made up of *m* bits b_j ($M = b_1 b_2 ... b_m$). The embedding factor, in bit/pixel, is $E_f = m / N$.

The original image is then divided in areas of size $\lfloor 1/E_f \rfloor$ pixels. Each area is used to hide **only one bit** b_j of the message. This splitting procedure guarantees that the message is

spread homogeneously over the whole *index* image. In order to hide the color palette, we need to embed 3 × 256 × 8 = 6144 bits (the number of colors is K=256) in the *index image*.

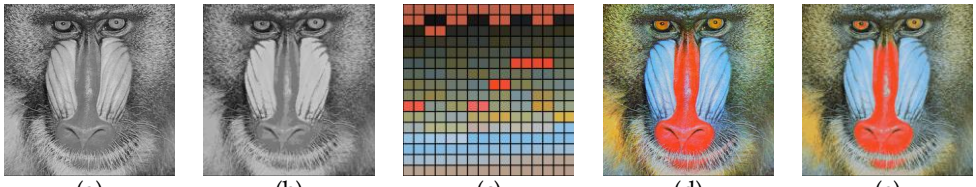
Consequently, the embedding factor E_f, only depends on the image size N. In our process, a Pseudo-Random Number Generator (PRNG) selects randomly, for each region, a pixel *Index(i)*. In order to get a marked pixel $Index_M(i)$, the LSB of this selected pixel *Index(i)* is then modified according to the message bit b_j (the formula is given for *index* values belonging to *[0,K-1]*):

$$Index_M(i) = Index(i) - Index(i) \bmod 2 + b_j,$$

This way to embed the color palette ensures that each marked pixel is at worst modified by one grey-level and in the same time that the rebuilt color pixel would not be very far from the right color value. Indeed, the third term of Equation (1) ensures that consecutive couples of color are close.

3.1.3. Evaluation of the fuzzy approach

In order to illustrate the performance of the fuzzy approach, we present results on few color images of size 256×256 pixels. For all the experiments, $\lambda_1 = 1$ and $\lambda_2 = 0.01 \times N/(K+1)$ (see Equation (1)). The obtained results show that the approach is efficient whatever the image type. Below, the different steps of the algorithm are commented on the baboon image.



(a) (b) (c) (d) (e)

Fig. 9. Application of the *fuzzy approach*: a) Luminance of the original color image, b) *Index* image, c) Color palette, d) Original color image, e) Rebuilt color image from the *index*-marked image.

After achieving the minimization of Equation (1) on the baboon image with *K=256* colors, we obtain an *index* image, illustrated in Figure 9.b, and its color palette illustrated in Figure 9.c. The luminance image of the original color image is given in Figure 9.a. One could observe the good similarity between *index* image and luminance image. The good PSNR value of 29.74 dB confirms this subjective feeling. We can notice that lots of *index* values are unused which explain the presence of some useless colors on the color palette of Figure 9.c. Also note that in the color palette in Figure 9.c, consecutive couples of color are colorimetricly close as expressed by the third term of Equation (1). Finally, we should precise that the computational cost for the minimization Equation (1) is very high. The article (Chaumont & Puech, 2007c) proposes a solution to reduce this cost but in counterpart, the results are less good.

The length of our embedded message (color palette) is *6144* bits which gives an embedding factor for an image of *256* × *256* pixels of E_f = *6144 / (256×256) = 0.093 bit/pixel*. The *index* image is then cut in block of *10* pixels. In each *10*-pixel block, **one** bit of the color palette is embedded at the position selected by the PRNG. The secured is obtained through the use of a secret key of 128 bits as a seed for the PRNG. The distribution of the message over the image is then key-related.

Figure 9.e shows the rebuilt color image from the *index*-marked one. This image is not visually far from the original color image even if the PSNR value of 27.90 dB is of middle quality. Note that the degradation due to the data-hiding method is weak because it disturbs *index* values of a maximum of one. This is made possible thanks to the color palette property to own consecutive couples of close colors.

PSNRs values are given in Table 3. Rebuilt color images are of middle quality (over 27 dB) but visually pleasant. Compared to the substitution-based approach of (Campisi et al., 2002) the rebuilt color images, with the fuzzy approach, are sometimes better and sometimes worse. On the other side, the *index* image is extremely close to the luminance image. We will see, in the next section, with the reordering approach, that if we reduce the strong constraint on the luminance, the rebuilt color images are of better quality.

PSNR values for *index*-marked images are over 29 dB. Since the grey-level images (i.e. the *index*-marked images) are of good quality, images may be colorized semi-automatically (Chaumont & Puech, 2008). This colorization may be seen as an indirect attack since an attacker is able to retrieve a pleasant color image and then fraudulently use it. Next section will expose a solution where *index* image have a weak quality and the rebuilt color image of have a better quality.

Images	PSNR *(luminance, index-marked)*	PSNR *(original color, rebuilt)*
baboon	29.74 dB	27.90 dB
barbara	34.86 dB	30.74 dB
peppers	35.03 dB	31.68 dB
airplane	35.95 dB	33.66 dB
lena	37.87 dB	34.78 dB
house	35.40 dB	35.45 dB

Table 3. PSNR between the luminance image and the *index*-marked image and PSNR between the original color image (RGB space) and the rebuilt color image (RGB space) using the *index*-marked image embedding its color palette.

3.2. The reordering approach

3.2.1. The image decomposition

Reducing the color number of a color image is a classical quantization problem. The optimal solution, to extract the *K* colors, is obtained by solving:

$$\{P_{i,k}, C(k)\} = \arg \min_{P_{i,k}, C(k)} \sum_{i=1}^{N} \sum_{k=1}^{K} P_{i,k}.dist^2(I(i), C(k)),\ and\ \forall i, \exists! k', \left[P_{i,k'} = 1\ and\ \forall k \neq k', P_{i,k} = 0\right], \quad (2)$$

where *I* is a color image of dimension *N* pixels, *C(k)* is the k^{th} color of the research *K* colors, *dist* is a distance function in the color space (L2 in the RGB color space), and $Pi,k \subset \{0, 1\}$ is the membership value of pixel *i* to color *k*.

A well known solution to minimize the Equation (2), and then to obtain the *K* colors, is to use the ISODATA k-mean clustering algorithm (Ball & Hall, 1966). $P_{i,k}$ is defined as:

$$\forall i, \forall k, P_{i,k} = \begin{cases} 1 \text{ if } k = \arg\{\min_{k' \in [1,K]} dist(I(i), C(k'))\}, \\ 0 \text{ otherwise,} \end{cases}$$

with $C(k) = \frac{\sum_{i=1}^{N} P_{i,k} \times I(i)}{\sum_{i=1}^{N} P_{i,k}}$.

Nevertheless, in our approach the *K* number is significant (*K=256*). If we proceed with a classical k-mean algorithm, the number of colors extracted will often be below *K*. Indeed, it is the well known problem of *death classes*. Moreover, the k-mean algorithm is quite long in CPU time in comparison to non optimal but faster algorithms such that *octree color quantization algorithm* of Gervautz and Purgathofer (Gervautz & Purgathofer, 1990), *Median Cut algorithm* (Heckbert, 1982)... To overcome those two problems ("death classes" and "CPU time"), we are using the *octree color quantization algorithm* as an initialization to the k-mean algorithm: $P_{i,k}$ are set from the result obtained with the *octree color quantization algorithm*.

3.2.2. The Layer Running Algorithm

Once the color quantization has been processed, the obtained *K* color image could be represented by an *index image* and a color palette. The *index* image is noted down *Index* and is defined such that:

$$\forall i \in [1, N], Index(i) = \arg \max_{k \in [1,K]} P_{i,k}.$$

The color palette is noted down *Palette* and $\forall k \in [1, K], Palette(k) = C(k)$.

Our goal is to get an *index* image where each grey-level is not too far from the luminance of the original color image. A second weakest constraint is that in the color palette, two consecutive colors should be close. Thanks to the color quantization, we already own an *index* image and a color palette. Our problem is then to find a permutation function which permutes in the same time the values of the *index* image and the values of the color palette. The best permutation function Φ is found by solving:

$$\Phi = \arg\min_{\Phi} \sum_{i=1}^{N} dist^2(Y(i), \Phi(Index(i))) + \lambda \sum_{k=1}^{K-1} dist^2(Palette(\Phi^{-1}(k)), Palette(\Phi^{-1}(k+1))), \quad (3)$$

where *Y* is the luminance of the original color image, and λ a constant value. The Φ permutation function is a bijective function in ℕ defined such that Φ: [1..K] → [1..K].

In a first approximation, Equation (3) is solved thanks to an heuristic algorithm: the *layer running algorithm* (Chaumont & Puech, 2007a). The aim of this algorithm is to find an ordering for the *K* colors such that consecutive colors are close and such that colors are

ordered from the darkest to the lightest. This ordering defines for each k^{th} color a k' position which gives us the Φ function such that $\Phi(k) = k'$.

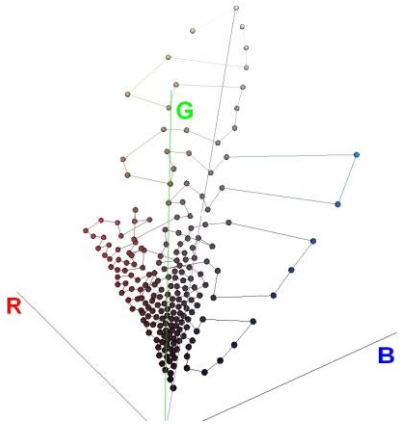


Fig. 10. A view of the layer running in the RGB cube.

To find an ordering of the *K* colors, the algorithm runs the color space to build the ordered suite of colors, illustrated Figure 10. This running is obtained by jumping from color to color, into the color space, by choosing the closer color from the current one. The first color of this suite is chosen as the darkest one among the *K* colors. An additional constraint to this running is that we limit the color research to colors which are very close in luminance. This signifies that the running in the color space is limited to a layer defined on luminance information. This *layer running algorithm* could then be seen as a kind of *3D spiral run* in the color space. A description of the algorithm may be found in (Chaumont & Puech, 2007a).

This *layer running algorithm* owns an implicit hidden parameter which is the *layer size* used during the color running in the color space. Since our goal is to minimize the Equation (3), a satisfying way to automatically set this parameter is to test all the possible values for this *layer size* parameter and to keep the *layer size* value minimizing the equation. Knowing that the possible values of the *layer size parameter* belong to the range [1, K] and that it is very fast to make just one run in the color space, this gives an elegant and fast solution to approximate the Equation (3).

Another problem still unsolved is the tuning of the lambda parameter. The equation below gives more details on the way lambda is expressed: $\lambda = \alpha \times N / (3 \times (K-1))$, with α the value balancing the two constraints evoke above and expressed by Equation (3). For example, α value set to *1* means giving the same weight to the two constraints, an α value set to *0.5* signifies an *index image* nearest to the luminance image, *a contrario* an α value set to *2* means a color palette more continuous.

3.2.3. Spatial data hiding

Similarly to the fuzzy approach, we embed the color palette information by modifying pixels of 0 or +1 or – 1. The embedding approach is also substitutive but it is not a simple

LSB approach. The objective is again to embed a message M made up of m bits b_j ($M = b_1b_2...b_m$). The original image is divided in areas of equal sizes and each area is used to hide **only one bit** b_j of the message.

A Pseudo-Random Number Generator (PRNG) selects randomly, for each region, a pixel *Index(i)*. In order to get a marked pixel $Index_M(i)$, the selected pixel *Index(i)* is modified according to the message bit b_j:

$$Index_M(\imath) = \begin{cases} Index(\imath) \text{ if } b_j = Index(\imath) \bmod 2, \\ \arg\min_{k \in \{Index(\imath)-1, Index(\imath)+1\} \cap \{1,\ldots,K\}} (Palette(Index(\imath)) - Palette(k))^2 \text{ otherwise.} \end{cases}$$

Thus, the *index* value *Index(i)* is modified of +1 or -1 when $b_j \neq Index(i) \bmod 2$. The best choice for this modification is then to choose the closest color between *Palette(Index(i)+1)* and *Palette(Index(i)-1)* in order to minimize the distance to the color *Palette(Index(i))*. This way to embed the color palette ensures that each marked pixel is at worst modified by one grey-level and in the same time that the rebuilt color pixel would not be very far from the right color value.

3.2.4. Evaluation of the reordering approach

In order to illustrate the performance of the reordering approach, we present results on few color images of size 256×256 pixels. The obtained results show that the approach is efficient whatever the image type. Below, the different steps of the algorithm are commented on the baboon image.

(a) (b) (c) (d) (e)

Fig. 11. Application of the *layer running algorithm*: a) Luminance of the original color image, b) *Index* image after color ordering, c) Color palette after color ordering, d) Original color image, e) Rebuilt color image from the *index*-marked image.

After the fast quantization step (with *K=256* colors) and the optimized *layer running algorithm* step, we obtain an *index image* and its color palette shown in Figures 11.b and 11.c. Note that the *index* image (Figure 11.b) is more contrasted than the luminance of the original color image (Figure 11.a) but keeps its semantic intelligibility. Also note that in the color palette, Figure 11.c, consecutive colors are colorimetricly close.

In order to be able to recover the embedded color-palette at the receiver, the embedding factor for an image of 256×256 pixels, is set to $E_f = 6144 / (256×256) = 0.093$ *bit/pixel*. The *index* image after color ordering is then cut in block of *10* pixels. In each *10*-pixel block, **a** bit of the color palette is embedded at the position selected by the PRNG. The secured is

obtained through the use of a secret key of 128 bits as a seed for the PRNG. The distribution of the message over the image is then key-related.

Figure 11.e shows the rebuilt color image from the *index*-marked one. This image is visually near from the original quantified one and the $PSNR_{(quantized\ ,\ rebuilt)}$ of 42.3 dB confirms this feeling. Some PSNRs values are given in Table 4. We could notice that rebuilt color images are of good quality (over 33 dB). PSNR values for *index* images are below 20 dB which is in general a poor result but in our case, it helps us to counter-attack the colorization attack (Chaumont & Puech, 2008). Moreover, *index* images are still semantically understandable.

Images	PSNR *(luminance, index−marked)*	PSNR *(original color, rebuilt)*
baboon	16.75 dB	33.31 dB
peppers	19.76 dB	36.32 dB
lena	19.11 dB	38.63 dB
house	18.64 dB	39.27 dB
airplane	12.87 dB	39.9 dB

Table 4. PSNR between the luminance image and the *index*-marked image and PSNR between the original color image (RGB space) and the rebuilt color image (RGB space) using the *index*-marked image embedding its color palette.

4. The 512 colors into an 8-bit-grey-level image approach

In this section we present a new approach based on a reversible data-hiding method allowing us to hide a color palette of 512 colors in the *index* image (Chaumont & Puech, 2008). Compared to the previous work of (Chaumont & Puech, 2007a, Chaumont & Puech, 2007b) presented in Section 3, the three main contributions of this approach are:

- the used of 512 colors (*index* values belongs to [0, 511]; *index* image is thus a 9-bit coded image; and there are 512 colors in the palette) but a resultant grey-level image owning values belonging to [0, 255]. The stored image (grey-level image) is then a classical 8-bit grey-level image and embeds a color palette plus a bitplane.
- the compression of a bitplane and the color palette,
- and the use of a high capacity reversible watermarking algorithm.

The overview of the embedding/extraction scheme is given in Figure 12. During the embedding scheme, the color image is quantized in a 512 color image. The 9-bit *index* image is split into an 8-bit grey level image and a bitplane. The extracted bitplane and the color palette are then embedded in the 8-bit grey level image.

The quantization of a color image into an *index* image and a color palette is achieved similarly to the color image decomposition detailed in Section 3.2.1. One applies the *octree color quantization algorithm* of Gervautz and Purgathofer (Gervautz & Purgathofer, 1990) follows-up with a k-mean clustering (Ball & Hall, 1966). Note that we decompose the image in 512 colors and not in 256 colors.

The palette re-ordering uses the *Layer Running Algorithm* which is detailed in Section 3.2.2. This re-ordering produces an *index* image visually similar to the luminance image and also

give a color palette whose consecutive color values are very close. Note that those two properties are necessary for the compression and data hiding parts.

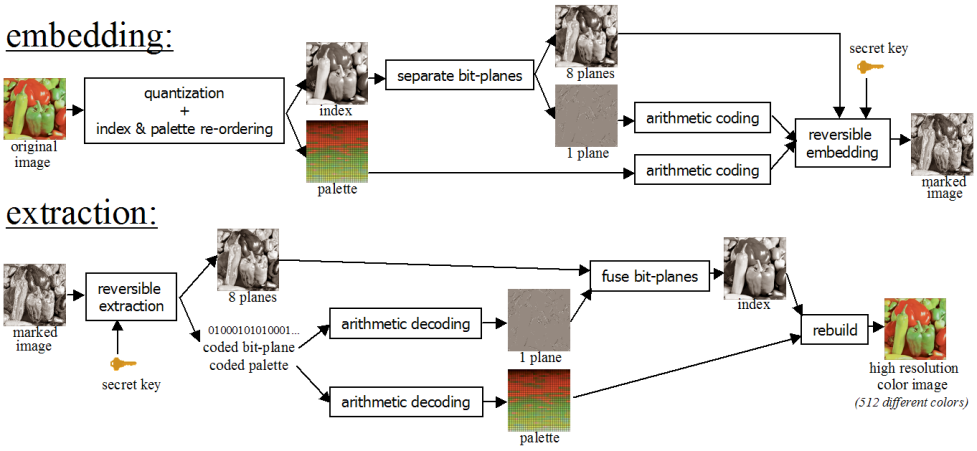


Fig. 12. The general embedding/extraction scheme.

Once *index* image and color palette are computed and re-ordered, the data hiding is achieved. Section 4.1 explains what is the message to embed, how to format it and how the reversible-watermarking algorithm works. After the watermarking process, the resultant grey-level image may be stored on a public website. The reversible extraction scheme, illustrated in Figure 12, is the exact inverse of the embedding scheme.

4.1. Data-hiding

4.1.1. Message construction

Once the color image decomposition (into 512 colors) and its re-arrangement (*layer running algorithm*) have been achieved, one bitplane of the *index* image is removed in order to create the cover image. The extracted bitplane and the color palette are then compressed in order to be embedded into the cover image (see Figure 12). The solution retained to compress the color palette is to statistically encode the error of prediction (differential encoding + arithmetic encoding) of each color belonging to the color palette.

The compression rate of the chosen bitplane is not enough efficient if directly achieved. One has to modify the statistics in order to reduce the entropy (theoretical rate cost). Thus, the bitplane vector (noted **bp**) is transformed in a prediction error vector (noted **e**) owning three possible values: *{-1, 0, 1}*. Each bit **bp**(i), $i \in [0, N]$, of the bitplane vector is then substituted by **e**$(i) \in \{-1, 0, 1\}$ before the lossless coding.

For each pixel $I(i)$ at position i, we first compute a prediction $Pred_{pixel}(i)$:

$$Pred_{pixel}(i) = \begin{cases} C \; if \; |A-B| < |A-C| \\ B \; otherwise, \end{cases}$$

with A, B and C the neighbors pixels of current pixel *I(i)* as defined below:

previous line	*A*	*B*
current line	*C*	*I(i)*

This kind of prediction is used in the Differential Pulse Code Modulation (DPCM) approach. It essentially takes into account the edges in order to predict more cleverly than with a simple mean. Note that some more sophisticated predictions may be explored in particularly by looking at lossless compression techniques. Nevertheless, this predictor is enough efficient for our approach. Also note that for image border the prediction is achieved with the available values.

The second step is to find the best prediction for the bit **bp**(i) of the bitplane vector. At the coder and the decoder sides, the information of all the other bitplanes is known. Thus, one have two possible prediction solutions: either predict a 0 knowing the other bitplanes, either predict a 1 knowing the other bitplanes. Knowing the prediction $Pred_{pixel}(i)$, one have to decide which is the best hypothesis: having a 0 in the bitplane (in this case the pixel value is noted $I_0(i)$) or having a 1 (in this case the pixel value is noted $I_1(i)$). This best hypothesis (best prediction) is chosen in accordance to the maximum of the two probabilities $p(Pred_{pixel}(i) \mid I_0(i))$ and $p(Pred_{pixel}(i) \mid I_1(i))$. This is a classical two hypothesis choice and with Gaussian pdf assumptions the best choice is:

$$Pred_{bit}(i) = \begin{cases} 0 \; if \; (Pred_{pixel}(i) - I_0(i))^2 < (Pred_{pixel}(i) - I_1(i))^2 \\ 1 \; otherwise. \end{cases}$$

The last step in order to obtain the **e** prediction error vector is to compute the prediction error (error values are either -1, 0 or 1):

$$\mathbf{e}(i) = \mathbf{bp}(i) - Pred_{bit}(i).$$

The prediction **e** error vector may now be encoded and because of the good prediction behaviour, the entropy is low and then the encoding cost is low.

4.1.2. Reversible watermarking scheme

The algorithm used for the reversible embedding is one of the three most efficient for its embedding capacity. A brief description is given below; for more details see (Chaumont & Puech, 2009). The algorithm lies on congruence computations. The congruence properties allow defining **three possible states** for a pixel:

- the state ***embedding*** which corresponds to a pixel *embedding* an integer coefficient belonging to *[1, n]*,
- the state ***to-correct*** which corresponds to a pixel that has been modified but *does not embed* any information; this pixel will be *corrected* during the reverting process,
- the state ***original*** which corresponds to an *original* pixel (i.e unchanged).

Let's define a constant integer value ***n*** greater or equals to 3. Let's also define the T transform which takes two integers x_1 and x_2 as input and return an integer:

$$T : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$
$$T(x_1, x_2) = (n+1).x_1 - n.x_2.$$

We will name the coding process, the act of embedding a message into an image and the decoding process, the act of extracting the message and rebuilding the original image. Let's now define the three possible states and the coding-decoding algorithms.

Embedding state

A pixel *i* in the *embedding* state is a pixel such that:

$$0 \leq T(I(i), I(i+1)) \quad and \quad T(I(i), I(i+1)) + n \leq L, \tag{4}$$

with *I* the original image (of *N* pixel size) whose grey-levels belongs to *[0,L]*. In the case of 8-bits images, *L* equals to *255*. All pixels *i* in the *embedding* state:

- are T transformed such that $I_T(i) = T(I(i), I(i+1))$, with $I_T(i)$ the resulting transformed pixel,
- **must** then embed **a** coefficient w **belonging to *[1, n]*** such that $I_w(i) = I_T(i) + w$.

Note that after the embedding of a coefficient w belonging to *[1, n]*, it is impossible to recover *I(i)* with the only knowledge of *I(i+1)*. Indeed $I_w(i) = (n+1).I(i) - n.I(i+1) + w$ with $w \neq 0$, thus $I(i) = \frac{I_w(i)+n.I(i+1)-w}{n+1}$ and so:

$$(I_w(i) + n.I(i+1)) \ mod \ (n+1) \neq 0. \tag{5}$$

The **congruence property** of Equation (5) allows detecting an **embedding** pixel during the decoding process. Note that during the decoding process, the $I_w(i+1)$ pixel should have been previously reverted to *I(i+1)* in order to compute this congruence. This implies that the image scan order used during the decoding process is the opposite of the one used during the coding process.

To-correct state

A pixel *i* in the *to-correct* state is a pixel such that the negation of Equation (4) is true:

$$T(I(i), I(i+1)) < 0 \quad or \quad T(I(i), I(i+1)) + n > L.$$

All pixels that are in this *to-correct* state are then modified such that:

$$\begin{aligned} &c \leftarrow (I(i) + n.I(i+1)) \ mod \ (n+1); \\ &if \ (I(i) - c) < 0 \ \ then \ \ c \leftarrow -(n+1-c); \\ &I_w(i) \leftarrow I(i) - c. \end{aligned} \tag{6}$$

The *c* coefficients belong to *[-n, n]* and are embedded (into the *embedding* pixels) in order to enable the reversibility of the *to-correct* pixels during the decoding process. We name the *c* coefficients the **corrective codes**. Note that after the modification expressed by Equation (6), pixel $I_w(i)$ checks the property:

$$(I_w(i) + n.I(i+1)) \ mod \ (n+1) = 0. \tag{7}$$

The **congruence property** of Equation (7) allows detecting a *to-correct* pixel at the reverting process. Note that at the decoding process the $I_w(i+1)$ pixels should have been previously reverted to $I(i+1)$ in order to compute this congruence.

Original state

Given an image order scan for the coding, a pixel in the *original* state (i.e. unmodified pixel) must always be present just before a pixel in the *to-correct* state. For a top-bottom-left-right scan order, if a pixel at position *i* is in the *to-correct* state, then the pixel at position *i-1* **must be** in the *original* state. In order to ensure this strong property (*original* and *to-correct* pixels go by pairs), during the scan, when a pixel at position *i* is detected as a *to-correct* one, a forward research is achieved in order to find the next *embedding* position (noted *next*). *Original* and *to-correct* states are then alternates between the *i* (or *i-1*) position and the *next-1* position.

This grouping constraint breaks the problematic dependencies during the decoding process. Remember that during the decoding, the image scan order is inverted. A *to-correct* pixel at position *i* may not be reverted immediately if its associated corrective code is still not extracted. Nevertheless, because the pixel at position *i-1* is an *original* pixel, the pixel at position *i-2* may be treated immediately and the decoding process may continue (pixel at position *i* will be corrected later, in a second pass).

Coding and decoding algorithms

The **coding** process is composed of two steps:

- classify each pixel in one of the three states: *embedding*, *to-correct*, *original*,
- embed into the *embedding* pixels, the watermark made of corrective codes plus the message.

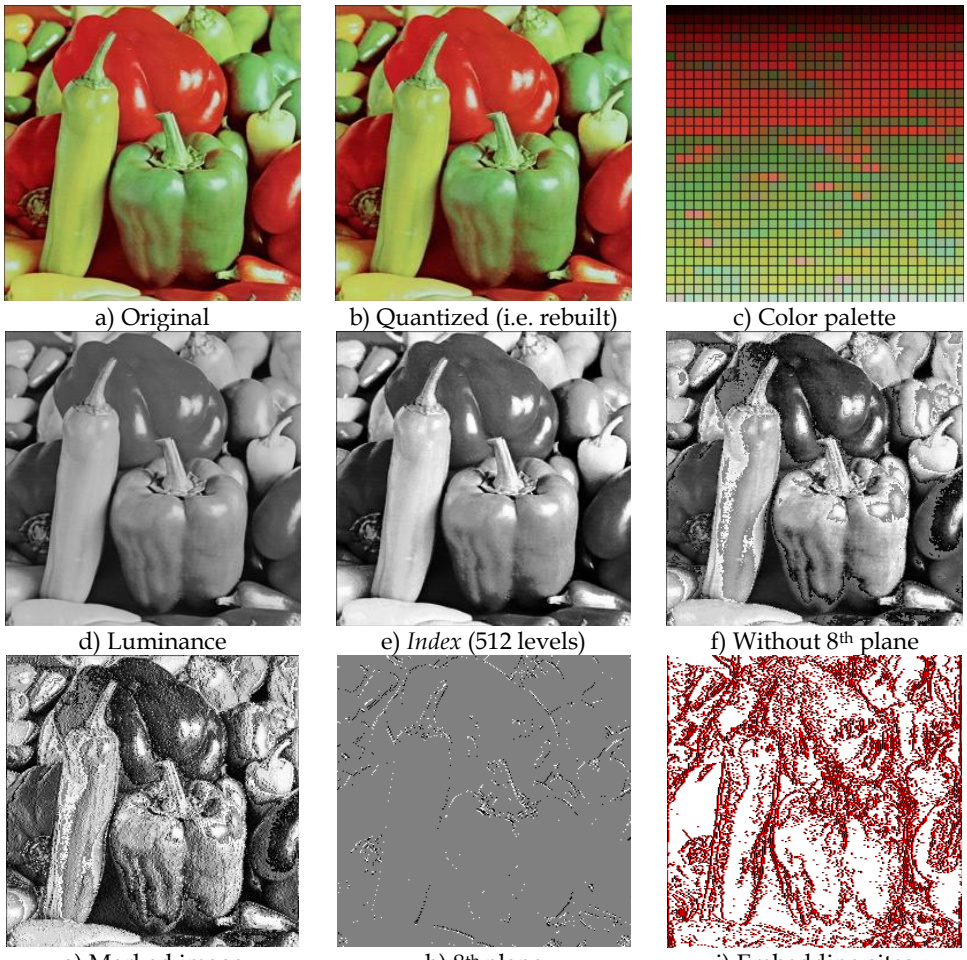
For the **decoding** process, the image scan order is inverted. The decoding process is also composed of two steps:

- extract the watermark from the *embedding* pixels, revert (during the scan) all those pixels and localize the *to-correct* pixels,
- from the extracted watermark retrieve the corrective codes and the message, and correct the *to-correct* pixels.

Note that the security is ensured by the secrecy of the scan order. The user secret key is used as a seed of a pseudo-random number generator. The obtained pseudo-random number sequence is used to generate an adapted scan order. Thus, no information about the message (color palette + bitplane) may be extracted by an attacker. Moreover, an attack by colorization (Chaumont & Puech, 2008), which consists in retrieving semi-automatically (a small human intervention is necessary) the colors of each pixels, is difficult since the final grey-level image has a poor quality. Also note, that this reversible watermarking scheme is not robust to any signal processing but it is not a deficiency for this application.

4.2. Evaluation of the 512 colors into an 8-bit-grey-level image approach

We have applied the "512 colors approach" on well known color images of size *256×256* pixels. For all the experiments, $n = 4$ for the reversible watermarking. The obtained results show that the approach is efficient whatever the image type. In Figure 13, the main steps of our approach are comment for the *peppers* image.



a) Original b) Quantized (i.e. rebuilt) c) Color palette

d) Luminance e) *Index* (512 levels) f) Without 8th plane

g) Marked image h) 8th plane i) Embedding sites

Fig. 13. Steps of the "512 colors" approach.

After achieving the color decomposition of the *peppers* image (Figure 13.a) we obtain an *index* image whose *index* values belongs to *[0, 511]* (Figure 13.e), and a color palette made of *K=512* colors (Figure 11.c). The quantized image, which is rebuilt with the knowledge of the *index* image and the color palette, is illustrated in Figure 13.b. The PSNR of this quantized image equals to 38.95 dB. The best PSNR obtained with a palette-based approach was 36.32 dB with the re-ordering one (Chaumont & Puech, 2007a). The obtained **gain** with the "512 colors" approach is more than to **2 dB** for the rebuilt color image.

The luminance image of the original color image is illustrated Figure 13.d. One could observe the good visual similarity between this luminance image and the *index* image. The *index* image is more contrasted but keeps its semantic intelligibility.

Once color image decomposition has been achieved, the *index* image is split into an 8-bit image (Figure 13.f) and a one bitplane image (a binary image). The chosen bitplane is the 8th for *peppers* image. The binary image is then modified in order to obtain a ternary error prediction image of very low entropy. This error prediction image is illustrated in Figure 13.h (in grey the 0 value, in black the -1 value, and in white the 1 value). The error prediction image and the color palette are then encoded with an arithmetic coder (Said, 2003). The embedded message is then made of the two bitstreams concatenations. Figure 13.i is a map giving the localization of the embedding site (white pixels) with the reversible watermarking approach of (Chaumont & Puech, 2009). Figure 13.g shows the final 8-bit image embedding a color palette (of 512 colors) and a bitplane image (the 8th bitplane of the *index* image).

The quality of the watermarked image (see Figure 13.g) is poor but as previously explains this is a good property. Indeed, this image yields difficult the colorization attack (Chaumont & Puech, 2008). On the contrary, the rebuilt color image (knowing the secret key) gives a very good quality color image (PSNR = 38.95 dB).

a) Original b) Watermarked c) Rebuilt (i.e quantized)

Fig. 14. Example on barbara image.

Another result is shown for *barbara 315×230* image in Figure 14. Figure 14.a is the original image, Figure 14.b is the watermarked image and Figure 14.c is the rebuilt (i.e quantized) image with a PSNR of 38.75 dB. Note that the semantic content is preserved in this watermarked image.

Images	Queiroz & Braun	Campisi et al.	Fuzzy	Re-ordering	512 colors
baboon	23.93 dB	29.8 dB	27.90 dB	33.31 dB	35.86 dB
peppers	28.82 dB	32.36 dB	31.68 dB	36.32 dB	38.95 dB
lena	30.31 dB	36.75 dB	37.87 dB	38.63 dB	40.93 dB
house	30.75 dB	31.76 dB	35.45 dB	39.27 dB	41.67 dB
airplane	28.56 dB	32.58 dB	33.66 dB	39.90 dB	42.96 dB

Table 5. PSNR between the original color image and the rebuilt color image.

In Table 5, few PSNRs are computed between the original color image and the rebuilt one. For all the images, the 512 colors approach is 2 dB greater than the second best approach of (Chaumont & Puech, 2007a).

5. Conclusion

In this chapter, we discussed about various solutions in order to protect the color information. First, we remind two quite far-off techniques usable for protecting the color (Queiroz & Braun, 2006, Campisi et al., 2002). Those techniques are based on wavelet sub-bands substitutions. They provide good solutions for a first approach. Second, we talk about two palette-based approaches: the re-ordering approach (Chaumont & Puech, 2007a) and the fuzzy approach (Chaumont & Puech 2007b). We conclude that the solution using the re-ordering layer running algorithm (Chaumont & Puech, 2007a) gives the best results in term of quality for the rebuilt color image, in term of security facing the colorization attack (Chaumont & Puech, 2008a) and in term of CPU computational complexity. Third, we present a new proposition: the 512 colors approach (Chaumont & Puech, 2008b).

The 512 colors approach consists to hide a 512 color palette in an 8-bit-grey-level image. The method is based on a decomposition of a color image in an *index* image and a color palette with 512 colors. The *index* image is then split in an 8-bit image and a binary image. The binary image and the color palette are then reversibly embedded in the 8-bit image. The resultant watermarked image is still semantically understandable.

Some objective PSNR measures are given and show that the 512 colors approach improves the general rebuilt color image quality compared to other approaches. Moreover, the security (only confidentiality is of interest: color retrieving should be difficult without the key) is reinforced by reducing the grey-level quality. Finally, the computational complexity is low.

In conclusion, protecting the color information providing a degraded *grey-level* image, semantically understandable, and a rebuilt color image of good quality is possible. Improvements are probably also possible by mixing the different approaches or with prior knowledge from both embedding and extracting sides.

6. References

Ball, G. H. & Hall, D. J. (1996). "ISODATA, A novel Method of Data Analysis and Pattern Classification," *in Proceedings of the International Communication Conference*, June 1966.

Campisi, P.; Kundur, D.; Hatzinakos, D. & Neri, A. (2002). "Compressive Data Hiding: An Unconventional Approach for Improved Color Image Coding," *EURASIP Journal on Applied Signal Processing*, Vol. 2002, No. 2, pp. 152–163, 2002.

Chaumont, M. & Puech, W. (2007a). "A Fast and Efficient Method to Protect Color Images," *in IS&T/SPIE 19th Annual Symposium on Electronic Imaging, Visual Communications and Image Processing, VCIP'2007, SPIE'2007*, vol. 6508, San Jose, California, USA, Jan. 2007.

Chaumont, M. & Puech, W. (2007b). "A Grey-Level Image Embedding its Color Palette," *in IEEE International Conference on Image Processing, ICIP'2007*, Vol. I, pp. 389–392, San Antonio, Texas, USA, Sept. 2007.

Chaumont, M. & Puech, W. (2007c). "Fast Protection of the Color of High Dimension Digital Painting Images," *in the 8th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'2007*, Santorini, Greece, 6-8 June 2007.

Chaumont, M. & Puech, W. (2008a), "Attack By Colorization of a Grey-Level Image Hiding its Color Palette," *in IEEE International Conference on Multimedia & Expo, ICME'2008*, Hannover, Germany, June 2008.

Chaumont, M. & Puech, W. (2008b). "A 8-Bits-Grey-Level Image Embedding its 512 Color Palette," *in the 16th European Signal Processing Conference, EUSIPCO'2008*, Lausanne, Switzerland, 25-29 August, 2008.

Chaumont, M. & Puech, W. (2009). "A High Capacity Reversible Watermarking Scheme, " *in IS&T/SPIE 21th Annual Symposium on Electronic Imaging, Visual Communications and Image Processing, VCIP'2009, SPIE'2009*, Paper 7257-54, San Jose, California, USA, 18–22 Jan. 2009.

Queiroz, De R. L. & Braun, K. (2006). "Color to Gray and Back: Color Embedding Into Textured Gray Images," *IEEE Transaction on Image Processing*, Vol. 15, No. 6, pp. 1464–1470, 2006.

Daubechies, I. & Sweldens, W. (1998). "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, Vol. 4, No. 3, pp. 247–269, 1998.

Dunn, J. C. (1974). "A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, Vol. 3, pp. 32–57, 1974.

Fridrich, J. (1998). "A New Steganographic Method for Palette-Based Images," *in proceedings of the IS&T PICS conference*, Apr. 1998.

Gervautz, M. & Purgathofer, W. (1990). "A Simple Method for Color Quantization: Octree Quantization," *Graphics Gems*, A.S. Glassner, pp. 287–293, 1990.

Heckbert, P. (1982). "Color Image Quantization for Frame Buffer Display," *Computer Graphics*, Vol. 16, No. 3, pp. 297–303, 1982.

Said, A. (2003), Chapter about Arithmetic Coding, In : *Lossless Compression Handbook*, Academic Press, 2003.

Tzeng, C.H.; Yang, Z.F. & Tsai, W.H. (2004). "Adaptative Data Hiding in Palette Images by Color Ordering and Mapping With Security Protection," *IEEE Transaction on Communications*, Vol. 52, No. 5, pp. 791–800, 2004.

Wu, M.-Y.; Ho, Y.-K. & Lee, J.-H. (2003). "An Iterative Method of Palette-Based Image Steganography," *Pattern Recognition Letters*, Vol. 25, pp. 301–309, 2003.