A Fast and Efficient Method to Protect Color Images

Chaumont M. and Puech W.

Laboratory LIRMM, UMR CNRS 5506, University of Montpellier II
161, rue Ada, 34392 MONTPELLIER CEDEX 05, FRANCE

ABSTRACT

In this paper, we propose a method to embed the color information of an image in a corresponding grey-level image. The objective of this work is to allow free access to the grey-level image and give color image access only if you own a secret key. This method is made of three major steps which are a *fast* color quantization, an *optimized* ordering and an *adapted* data hiding. The principle is to build an *index image* which is, in the same time, a semantically intelligible grey-level image. In order to obtain this particular *index image*, which should be robust to data hiding, a *layer running algorithm* is proceeded to sort the K colors of the palette. The major contributions of this paper are the *fast* color quantization, the *optimized* layer running algorithm, the color palette *compression* and the *adapted* data hiding.

Keywords: Data-Hiding, Color Image Secured, Color Palette, Compression.

1. INTRODUCTION

Nowadays, only few secure solutions are proposed in order to give both a free access to low-quality images and a secure access to the same images with an higher quality. Our proposed solution is built on a data-hiding method. The image may be freely obtained but its high quality visualization requires a secret key. More precisely, in our solution, a grey-level image is freely accessible but only secret key owners may rebuild the color image. Our aim is thus to protect the color information by embedding this information in the grey level image. Note that this work is though to be used to give a limited access to the private digital painting data-base of the Louvre Museum of Paris, France.

The originality of this paper is to build an *index image* which is, in the same time, a semantically intelligible grey-level image. In order to obtain this particular *index image*, which should be robust to data hiding, we propose an *optimized* K color ordering algorithm called the *layer running algorithm*. This paper carry on the previous work of Chaumont and Puech.[1] The new contributions are the *faster* quantization step, the *optimized* layer running algorithm, the *new* embedding step and the color palette *compression*.

Lots of works propose to hide information by using the decomposition of a color image in an *index image* and a color palette. The data-hiding may occur in the *index image*[2] or in the color palette.[3,4] Nevertheless, none of those technics tries to protect the color information i.e hide the color palette in the *index image*. Indeed, in our method we sort the colors of the color palette in order to get: **1.** an *index image* which is near of the luminance of the original color image and **2.** a color palette whose consecutive colors are close. By doing this, we obtain an *index image* and a color palette well adapted for the purpose of data-hiding and color information secured. Other works based on wavelet decomposition and sub-band substitution propose solutions to embed the chroma informations in a grey-level image.[5–7] Their areas are perceptive compression and image authentification for Campisi *et al*[5] and Zhao *et al*[6] and image printing for Queiroz and Braun.[7] Even if those techniques embed the color information, their approach and their purpose are clearly different from that exposed in that paper.

In section 2, we present the different steps of our method: the *fast* color quantization, the *optimized layer running algorithm*, the *new* data-hiding method and the color palette *compression*. In section 3, results are presented and compared to our previous proposed method.[1]

marc.chaumont@lirmm.fr; phone: +33 (0)4 67 41 85 14; william.puech@lirmm.fr; phone: +33 (0)4 67 41 86 85.

2. THE PROPOSED METHOD

2.1. Color quantization

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:

$$\begin{aligned}\{P_{\imath,k}, C(k)\} &= \arg\min_{P_{\imath,k}, C(k)} \sum_{\imath=1}^{N}\sum_{k=1}^{K} P_{\imath,k}.dist^2(I(\imath), C(k)), \\ &\quad and\ \forall \imath, \exists! k', P_{\imath,k'} = 1\ and\ \forall k \neq k', P_{\imath,k} = 0,\end{aligned} \tag{1}$$

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 $P_{\imath,k} \subset \{0,1\}$ is the membership value of pixel $\imath$ to color k.

A well known solution to minimize the Equ. (1), and then to obtain the K colors, is to use the ISODATA k-mean clustering algorithm.[8] $P_{\imath,k}$ is defined as:

$$\forall \imath, \forall k, P_{\imath,k} = \begin{cases} 1 \text{ if } k = \arg\{\min_{k' \in [1,K]} dist(I(\imath), C(k'))\}, \\ 0 \text{ otherwise,} \end{cases} \quad with\ C(k) = \frac{\sum_{\imath=1}^{N} P_{\imath,k} \times I(\imath)}{\sum_{\imath=1}^{N} P_{\imath,k}}.$$

Nevertheless, in our approach the K number is significant in comparison to the original number of colors. 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,[9] *Median Cut algorithm*[10]... 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: $P_{\imath,k}$ are set from the result obtained with the *octree color quantization algorithm*.

2.2. Layer running

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:

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

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

Our goal is to solve two constraints; the first constraint is to get an *index image* where each grey-level is not too far from the luminance of the original color image; the second 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:

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

where Y is the luminance of the original color image, and λ is the Lagrangian value. The Φ permutation function is a bijective function in $\mathbb{N}$ defined such that $\Phi : [1..K] \rightarrow [1..K]$.

In a first approximation, the Equ. (2) is solved thanks to an heuristic algorithm: the *layer running algorithm*.[1] 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 $\Phi(k) = k'$.

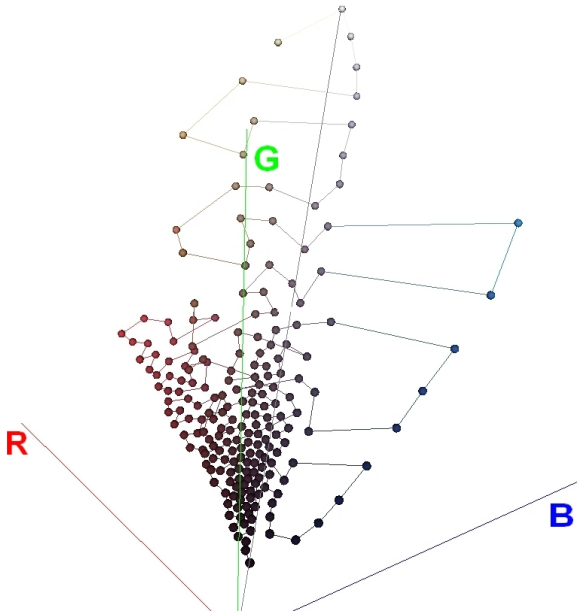


Figure 1. A view of the layer running in the RGB cube.

To find an ordering of the K colors, the algorithm runs the color space to build the ordered suite of colors, illustrated Figure 1. 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 not too far in luminance. This signify 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.

This *layer running algorithm* own 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 Equ. (2), 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 rapid solution to approximate the Equ. (2).

Another problem still unsolved is the tuning of the lambda parameter. The equation below gives more details on the way lambda is expressed:

$$\lambda = \alpha \times N/(3 \times (K-1)), \tag{3}$$

with α the value balancing the two constraints evoke above and expressed by Equ. (2). For example, an 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.

2.3. Spatial data hiding method

The methods in spatial domain embed directly the information into the pixel of the original image. The first techniques embedded the bit message in a sequential way in the LSB (Low Significant Bit) of the pixel image.[11,12] They have been improved by using a PRNG (Pseudo-Random Number Generator) and a secret key in order to have private access to the embedded information. The PRNG spreads over the image the message and makes hard the steganalyses.[13] Although those spatial hiding methods are not robust against attacks, they enable to embed a great amount of information. Note that the embedding process can be made more robust by using a more significant bits such the fourth bit of the pixel image.[14]

For this paper, we have used an algorithm to embed the color palette information in the LSB (Low Significant Bit) of the image of N pixels. The objective is thus to embed a message M made up of m bits $b_\jmath$ ($M = b_1 b_2 ... b_m$). 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** $b_\jmath$ of the message. This splitting procedure guarantees that the message is spread homogeneously over the whole image. Lets remark that when the color palette is **compressed**, one embed **less than** $3 \times 256 \times 8 = 6144$ 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, the PRNG selects randomly, for each region, a pixel $Index(\imath)$. In order to get a marked pixel $Index_M(\imath)$, the LSB of this selected pixel $Index(\imath)$ is then modified according to the message bit $b_\jmath$:

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

Thus, the index value $Index(\imath)$ is modified of $+1$ or -1 when $b_\jmath \neq Index(\imath) \ mod \ 2$. The best choice for this modification is then to choose the closest color between $Palette(Index(\imath)+1)$ and $Palette(Index(\imath)-1)$ in order to minimize the distance to the color $Palette(Index(\imath))$. This way to embed the color palette ensure 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.

2.4. Color palette compression

As explain above, an **uncompressed** color palette is a to-embed information that necessitates 6144 bits to be represented. One then compress this information in order to reduce the bit quantity to embed and to reduce the degradation over the *index image* and the rebuilt color image. Statistical algorithms (Shannon-Fano, Huffman or Arithmetic) are well adapted to reduce the coding cost but in the case of very small information quantity - which is the case here- the descriptive header tends to be costly in comparison to the source coding. Adaptive techniques or pre-learned tables which allow the suppression of this header are not satisfying in the case of very small information quantity. The solution is then to compress the description header adaptively to the problem of color palette compression.

The solution to compress the color palette is to encode statistically the **error of prediction** (differential encoding + entropy encoding) of each color belonging to the color palette. The Huffman algorithm has been used for the experiments. The descriptive header allowing to the decoder to rebuild the association between source symbols and codes is described on Figure 2.

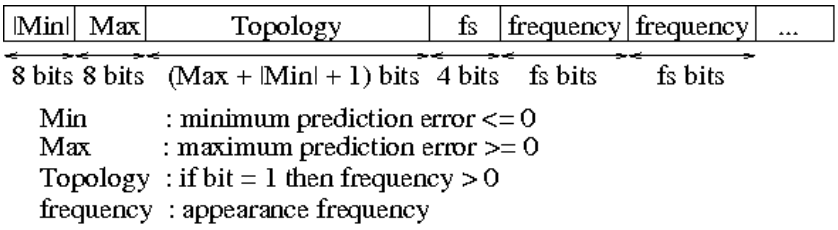
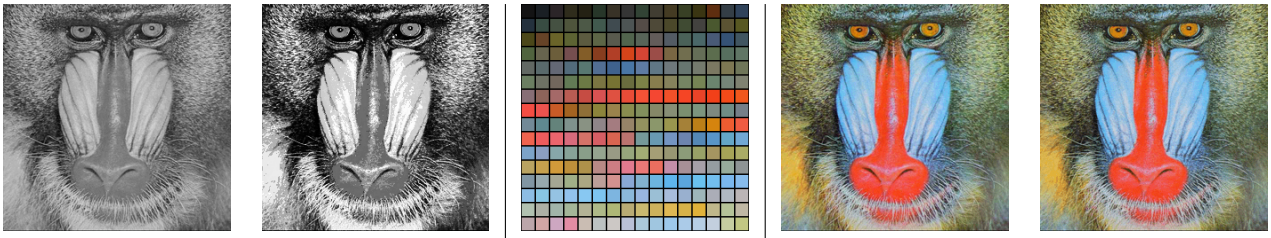


Figure 2. Descriptive header of the coded color palette.

The min (resp. max) value stands for the minimum (resp. maximum) value of the color **prediction errors**. A topological bit is set to 1 if the associated color error value is present. *fs* is the number of bits on which each frequency will be coded. The rest of the header is the frequencies of each present color error prediction. The originality of this header stands mainly on the topological size reduction. Most of the time, minimum (resp. maximum) value of the color error predictions are around -64 (resp. 64). With a minimum (resp. maximum) value of the color error predictions equals to -64 (resp. 64) one reduced the topological quantity of 356 bits (decrease of 511 bits to 155 bits).

3. RESULTS

We present the results of our method on three color images of size 256×256 pixels: baboon, airplane and house. The baboon image owns a lot of different colors, the airplane image owns very few colors and is light-colored and the house image owns tint areas. 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)

Figure 3. 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 show on Figures 3.b and 3.c. Note that the *index image* (Figure 3.b) is more contrasted than the luminance of the original color image (Figure 3.a) but keeps its semantic intelligibility. Also note that in the color palette, Figure 3.c, consecutive colors are colorimetricly close.

In comparison to our previous approach,[1] this new quantization method is really faster. For example, for the baboon 256×256, the quantization is 19 times* more rapid to get the same color PSNR quality of 33.7 dB†. Moreover, the layer size used in the *layer running algorithm* is automatically set in order to minimize the Equ. (2) with α set to 5 (see Equ. (3)). As a consequence the PSNR of the rebuilt color image is slightly improved because of a better color-palette ordering (see PSNR comparison on Table 1).

The length of our embedded message (color palette) after source coding and header computation gives a length message of $m = 5110$ bits. 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 keep to $E_f = 6144/(256 \times 256) = 0.093$ *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 as explain in Section 2.3. The secured is obtain through the used 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.

Thanks to the color palette compression there is a reduction in comparison to our previous approach[1] of the number of disturb index-pixels due to the data-hiding. The PSNR between the *index image* and the *index*-marked one is then increasing from 58.1 dB for[1] to 62.2 dB.

Figure 3.e shows the rebuilt color image from the *index*-marked one. This image is visually near from the original quantified one and the $\text{PSNR}_{(quantized,rebuilt)}$ of 42.3 dB confirms this feeling. The rebuilt color image is growing from 32.89 dB for our previous work[1] to only 33.31 dB. Nevertheless, there is a neat visual improvement due to the embedding technique. Note that the color palette compression and the optimized *layer running algorithm* are two other contributions which explain the improvement.

Few PSNR values are given on the Table 1 for the three images. One could remark that rebuilt color images are of good quality (over 33 dB). PSNR values for *index images* are under 20 dB which is in general a poor

*Experiments were proceed on an Intel Pentium(R) 3.2 GHz with 1GB RAM and it takes 1 min. versus 19 min. for our previous approach.[1]

†In Chaumont and Puech article,[1] quantization were obtained by running a Fuzzy c-mean algorithm and then a k-mean algorithm.

result but it is known that PSNR measure is not well adapted in case of strong contrast distortion. Indeed, the *index images* are visually pleasant. By comparing the results of our previous method[1] with the proposed one (Table 1), we could note a quality improvement of the rebuilt color images. Those improvements are due to the *layer running algorithm* **optimization**, the **adapted** data-hiding method and the color palette **compression**.

Table 1. PSNR comparisons.

images	$PSNR_{(luminance,index-marked)}$		$PSNR_{(color\ image,rebuilt)}$		color palette coding cost
	previous method[1]	proposed method	previous method[1]	proposed method	
baboon	16.31 dB	16.75 dB	32.89 dB	33.31 dB	header: 693 bits source: 4417 bits sum: 5110 bits
airplane	12.95 dB	12.87 dB	38.89 dB	39.90 dB	header: 761 bits source: 3552 bits sum: 4313 bits
house	18.56 dB	18.64 dB	38.07 dB	39.27 dB	header: 728 bits source: 3833 bits sum: 4561 bits

Other results on Figures 4 and 5 show the right behavior for house, airplane, peppers and lena image.

4. CONCLUSION

In this paper, we have proposed an improvement to a method hiding the color information in a grey-level image. This method is made of three major steps which are the color quantization, the color ordering and the data hiding. Those steps have been improved thanks to an acceleration of the quantization, a layer running algorithm optimization, a better data-hiding and color palette compression. The obtained results show a real improvement in complexity and in quality.

ACKNOWLEDGMENTS

This investigation was in part supported by the TSAR project which is a french national project ANR-05-SSIA-0017-05 of the ANR ARA SSIA (*Agence Nationale de la Recherche, Action de Recherche Amont, Sécurité Systèmes Embarqués et Intelligence Ambiante*).

We would like also to thank Mr Lahanier Christian of the C2RMF (*Centre de Recherche et de Restauration des Musées de France*) and the Louvre Museum for the digital paintings and for valuable discussions.

Finally, we thank Mr Allier Simon a student of the CUFR (*Centre Universitaire de Formation et de Recherche*) de Nîmes, France, for bringing together all the modules and for its Gervautz and Purgathofer[9] algorithm implementation.

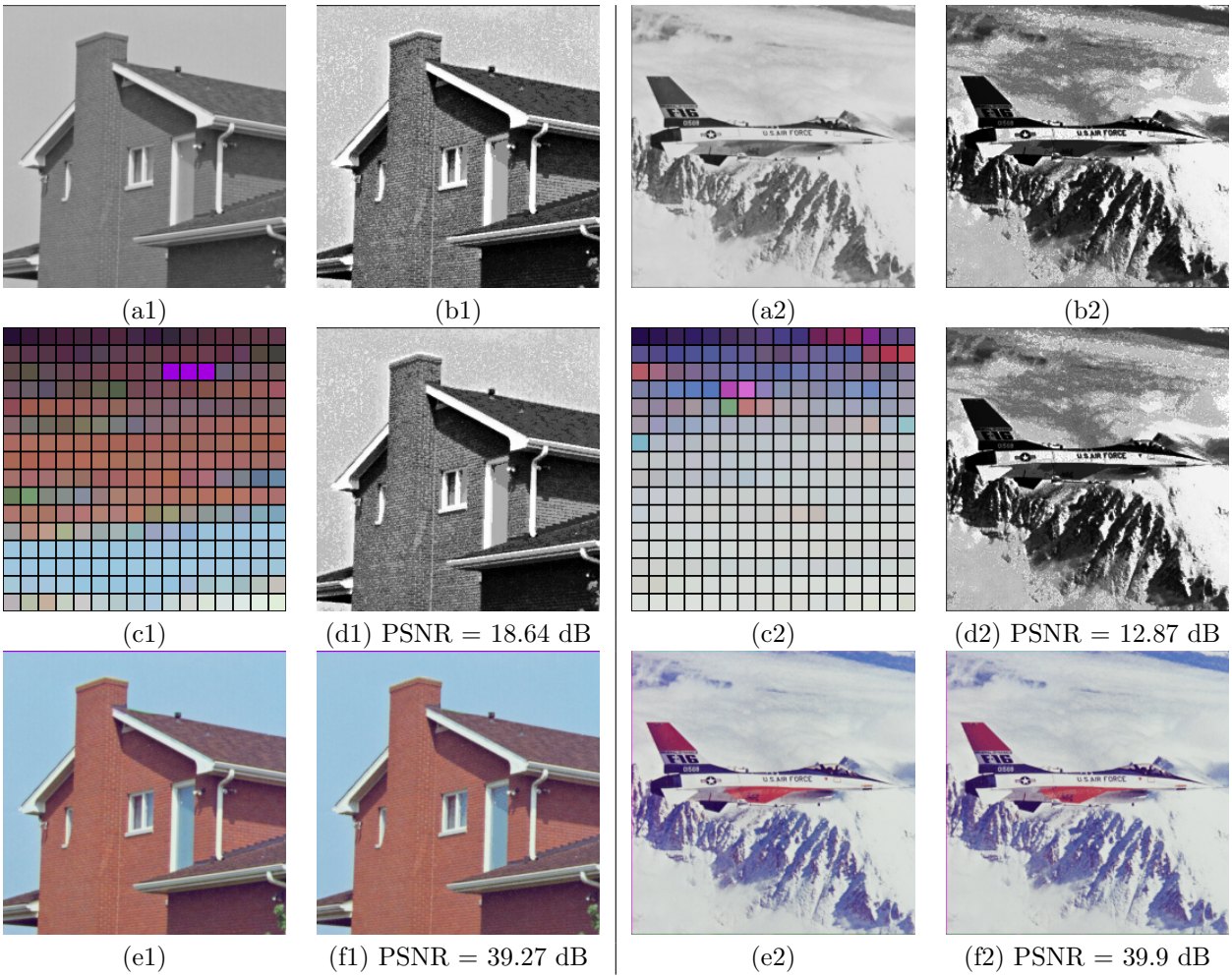


Figure 4. Application of the *layer running algorithm*: a1-a2) Luminance of the original color image, b1-b2) *Index* image after color ordering, c1-c2) Color palette after color ordering, d1-d2) *Index*-marked image, e1-e2) Original color image, f1-f2) Rebuilt color image from the *index*-marked image.

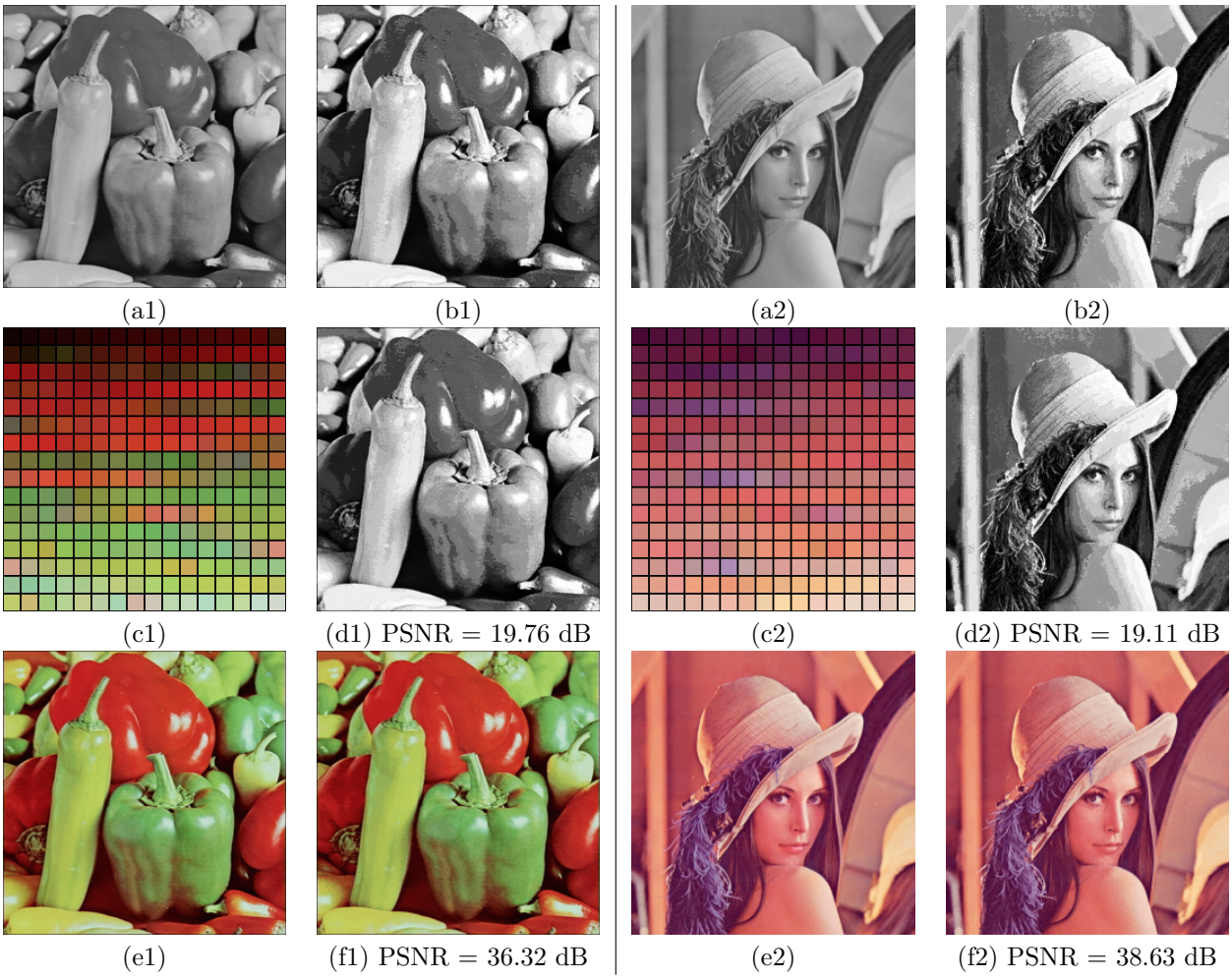


Figure 5. Application of the *layer running algorithm*: a1-a2) Luminance of the original color image, b1-b2) *Index* image after color ordering, c1-c2) Color palette after color ordering, d1-d2) *Index*-marked image, e1-e2) Original color image, f1-f2) Rebuilt color image from the *index*-marked image.

REFERENCES

1. M. Chaumont and W. Puech, "A Color Image in a Grey-Level Image," in *IS&T Third European Conference on Colour in Graphics, Imaging, and Vision, CGIV'2006*, pp. 226–231, (Leeds, UK), June 2006.
2. J. Fridrich, "A New Steganographic Method for Palette-Based Images," in *Proceedings of the IS&T PICS conference*, Apr. 1998.
3. M.-Y. Wu, Y.-K. Ho, and J.-H. Lee, "An Iterative Method of Palette-Based Image Steganography," *Pattern Recognition Letters* **25**, pp. 301–309, 2003.
4. C. Tzeng, Z. Yang, and W. Tsai, "Adaptative Data Hiding in Palette Images by Color Ordering and Mapping With Security Protection," *IEEE Transaction on Communications* **52**(5), pp. 791–800, 2004.
5. P. Campisi, D. Kundur, D. Hatzinakos, and A. Neri, "Compressive Data Hiding: An Unconventional Approach for Improved Color Image Coding," *EURASIP Journal on Applied Signal Processing* **2002**(2), pp. 152–163, 2002.
6. Y. Zhao, P. Campisi, and D. Kundur, "Dual Domain for Authentication and Compression of Cultural Heritage Images," *IEEE Transaction on Image Processing* **13**(3), pp. 430–448, 2004.
7. R. de Queiroz and K. Braun, "Color to Gray and Back: Color Embedding Into Textured Gray Images," *IEEE Transaction on Image Processing* **15**(6), pp. 1464–1470, 2006.
8. G. H. Ball and D. J. Hall, "ISODATA, A novel Method of Data Analysis and Pattern Classification," in *Proceedings of the International Communication Conference*, June 1966.
9. M. Gervautz and W. Purgathofer, "A Simple Method for Color Quantization: Octree Quantization," *Graphics Gems, A.S. Glassner* , pp. 287–293, 1990.
10. P. Heckbert, "Color Image Quantization for Frame Buffer Display," *Computer Graphics* **16**(3), pp. 297–303, 1982.
11. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding," *I.B.M. Systems Journal* **35**(3-4), pp. 313–336, 1996.
12. N. Nikolaidis and I. Pitas, "Robust Image Watermarking in the Spatial Domain," *Signal Processing* **66**(3), pp. 385–403, 1998.
13. J. Fridrich and M. Goljan, "Practical Steganalysis: State-of-the-Art," in *Proceeding of SPIE Photonics West, Electronic Imaging, SPIE'2002*, **4675**, pp. 1–13, 2002.
14. J. Rodrigues, J. Rios, and W. Puech, "SSB-4 System of Steganography using Bit," in *5th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'2004*, (Lisboa, Portugal), Apr. 2004.