Chapter 6
Copyright Protection of Palette Images by A Robust Lossless Visible Watermarking Technique

6.1 Overview of Proposed Method

A method for robust lossless visible watermarking of palette images for copyright protection against removal attacks is proposed in this chapter. In our local authentication centers mentioned previously, visible watermarks can be embedded into images before posting the images on the Internet. Users may obtain the embedded images for free, or they can request local authentication centers to remove embedded visible watermarks and acquire the original images with an additional fee.

In Section 6.1.1, the idea of the proposed lossless visible watermarking technique is described. Some properties of palette and graphics-interchange-format (GIF) images are introduced in Section 6.1.2. And in Section 6.1.3, we briefly describe the principle of the proposed method. In Section 6.2, the embedding and lossless recovery processes of the proposed method, the robustness against removal attacks, and the detailed algorithms are described. And in Section 6.4, some experimental results are shown. Finally, in Section 6.5, some discussions and a summary are made.

6.1.1 Idea of Proposed Robust Lossless Visible Watermarking Technique

In order to protect copyright of images, digital watermarking techniques, which include invisible watermarking and visible watermarking, are proposed for embedding a pre-defined watermark into an image. Traditionally, the processes of invisible watermark and visible watermark embedding usually result in image distortions. Even though the distortions are usually tiny, in some applications, such as medical and military, such distortions are unacceptable due to the high precision requirement in these applications. This intensifies the demand for lossless watermarking techniques, which are able to recover exactly the original images.

In addition to specific applications' requirements, lossless watermarking techniques can save image owners a lot of storage space. The owner can preserve only the stego-images instead of both the original images and their stego-versions, because by using lossless watermarking techniques they can obtain distortion-free original images from stego-images. Most of the lossless watermarking techniques focus on invisible watermarking [2]-[5], and few papers are found on lossy visible watermarking. A study is found in Hu *et al.* [9]. To the best of our knowledge, there is no paper on lossless visible watermarking of palette images so far.

Typical visible watermarking techniques were proposed for reducing the distortions of stego-images, so semi-transparent watermarks are embedded traditionally. However, we choose to embed opaque watermarks into cover images in this study. Although an opaque watermark results in more distortions visually than a semi-transparent watermark, but on the contrary, it has the advantages of better advertising effects and copyright declaration, and can be removed entirely by the proposed lossless watermarking technique when the image owner requires to get the original image.

6.1.2 Properties of Palette and GIF Images

A palette, in computer graphics, is a designated subset of the total range of colors supported by a computer graphic system. Each color in the palette is assigned a number, which is called an index number. This index determines the color of a pixel. A GIF image is an 8-bit palette image, and a portable–network-graphics (PNG) image can be an 8-bit-palette-based image, too.

The GIF image is adopted as the color palette image in our study. Before describing the method for the lossless watermarking technique for GIF images, some properties must be addressed. First, every GIF image contains at most 256 colors. Second, a color palette is used to store colors used in the image. Third, every pixel can be thought as an index number which is a reference to the color palette, so each pixel needs one byte of storage space. Finally, the GIF image uses a lossless compression algorithm, so there is no loss of any information when saving a GIF image.

6.1.3 Principle of Proposed Method

For the comprehensibility and the convenience to the following descriptions, some definitions are given first. The area in a cover image where a watermark is embedded will be called the *watermark area* and the remaining portion in the cover image will be called the *non-watermark one*. The pixels in the watermark with values 1 will be called *black watermark pixels*, denoted as W_b, and those with values 0 called *white ones* and denoted as W_w. The watermark is supposed to be embedded in the upper left corner of the image. The corresponding pixels of the cover image in the watermark area where W_b is supposed to be embedded will be called *black watermark* *embedded pixels* I_{Wb}, and the remaining portion in the watermark area will be called *white ones* I_{Ww}.

The basic idea of the proposed method is to replace the colors of I_{Wb} with other visually different colors and this replacement must be revertible. That is, to enable the embedded watermark looking visually different from the pixels adjacent to them and able to be removed losslessly with a right key. In order to make the embedded watermark look obviously different, we try to find the farthest color among the palette colors to replace with. In addition, we apply a two-level key protection to ensure the embedded visible watermark remaining visible when illegal users try to remove it without a right key.

Each visible watermark used for image copyright protection is assumed to be a binary image in this study. We first divide the cover image into a watermark area and a non-watermark area. We are only concerned about the watermark area. The watermark area is further divided into black watermark embedded pixels I_{Wb} which are going to be replaced with other colors, and white watermark embedded pixels I_{Ww} which will be kept unchanged. Then we count the occurrences of colors of I_{Ww}, and let the occurrences of color C_0 through C_{255} be denoted as O_0 through O_{255}. We sort the color palette by the occurrences O_i in a descending order. We call this sorted result a *sorted color palette* P_s. We propose an adjusted Euclidean distance by taking the concept of weighting into consideration, which is called a *weighted Euclidean distance*. A *weighted Euclidean distance* of color C_i is calculated by summing the products of the Euclidean distance between C_i and the other colors C_0 through C_{255} and the occurrences O_0 through O_{255}. The Euclidean distance between two colors C_1 and C_2 is defined as

$$\mu(c_1, c_2) = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}\ . \qquad (6.1)$$

And the weighted Euclidean distance of color C_i in the color palette is defined as

$$\mu(c_i) = \frac{\sum_{n=0}^{255} (Occurrence\ (c_n) \times \mu(c_i, c_n))}{\sum_{n=0}^{255} Occurrence\ (c_n)}. \qquad (6.2)$$

The function Occurrence(C_n) is the occurrence O_n of the color C_n counted only in colors of white watermark embedded pixels I_{Ww}. Taking this function as a weight helps us to find the farthest color among colors of I_{Ww} instead of the whole cover image.

Second, we find the farthest color from either the original color palette or the sorted color palette. What we want to choose is the color that has the largest weighted Euclidean distance. Using the weighted Euclidean distance helps us to find the farthest color among the colors of I_{Ww} instead of the whole cover image. We denote this farthest color by C_f, which is supposed to be the most different color from the colors of I_{Ww}. If the color of I_{Wb} is mostly replaced with C_f, the embedded watermark will look obviously different from its neighboring pixels, said I_{Ww}. After finding color C_f, we can set up a table beginning from the color in the palette which is closest to C_f to the color which is farthest to C_f. The result can be thought as a new color palette which has a different order from the original one. And we call this a *rearranged color palette* P_r.

Now we have a sorted color palette P_s and a rearranged color palette P_r. We can make a one-to-one mapping between the two palettes in their original order. That is, if the original color of the pixel ranks second in P_s, we map the color to the second color in P_r. Then we can replace all black watermark embedded pixels by this color mapping rule, and thus obtain a stego-image. As a consequence, we make most part of colors of I_{Wb} be replaced with a group of similar colors which are visually different from the colors of I_{Ww}.

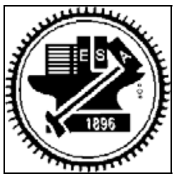
Finally, in order to prevent illegal users from removing the embedded watermark easily, we apply a two-level key protection when doing the color mapping. We add an offset generated by a secret key to the original order so that an illegal user cannot obtain the losslessly recovered image with a wrong offset generated by a wrong key. Moreover, there will be a noise-residual watermark on the recovered image. In addition to adding an offset, we randomize the position of the embedded watermark by the same key. The randomization results in an opaque embedded watermark, which has the advantage of better advertising effects and copyright declaration. The owner's copyright of the image can be protected further by this way.

6.2 Proposed Method for Copyright Protection of GIF Images

In Section 6.2.1, the visible watermark embedding process is introduced. And we describe the lossless recovery process of original images in Section 6.2.2. A noise-residual recovery process against removal attacks is presented in Section 6.2.3. Finally, detailed algorithms of embedding visible watermarks and lossless recovery of original images are described in Section 6.2.4.

6.2.1 Embedding of Removable Visible Watermarks by A Neighborhood-Dependent Watermarking Technique

In this section, the details of the proposed watermark embedding process are introduced. At first, we divide the watermark area into black watermark embedded pixels I_{Wb} and white ones I_{Ww}. Figure 6.1 shows an example of this.


(a)

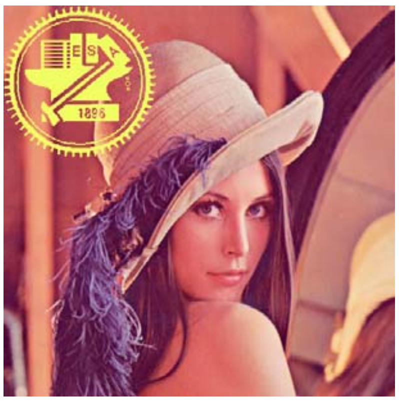
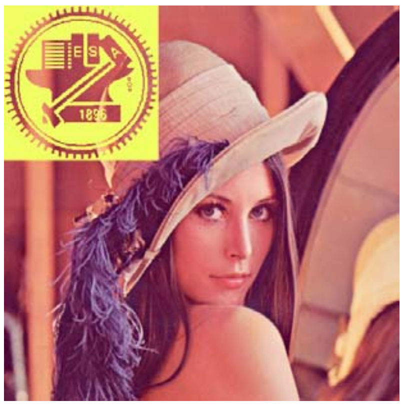


(b) (c)

Figure 6.1 An example of black watermark embedded pixels and white watermark embedded pixels. (a) Binary watermark. (b) Area of black watermark embedded pixels. (c) Area of white watermark embedded pixels.

And then we count the occurrences of the colors of white watermark embedded pixels I_{Ww}. A *sorted color palette* P_s can be set up by the occurrences in a descending order. The color which occurs with the highest frequency is ideally supposed to be replaced with a visually obviously different color. So we choose the farthest color from either the original color palette or the sorted one by picking a color C_f with a largest weighted Euclidean distance, and C_f will be the obviously different color to be replaced with. Once we have color C_f, we can find the closest color to C_f , which is called C_1, among the palette which we use to find color C_f by calculating the Euclidean distance between two involved colors. So we can set up a table from the closest color to the farthest color to C_f. We call this new table a *rearranged color*

Chapter 6
Copyright Protection of Palette Images by A Robust Lossless Visible Watermarking Technique

6.1 Overview of Proposed Method

A method for robust lossless visible watermarking of palette images for copyright protection against removal attacks is proposed in this chapter. In our local authentication centers mentioned previously, visible watermarks can be embedded into images before posting the images on the Internet. Users may obtain the embedded images for free, or they can request local authentication centers to remove embedded visible watermarks and acquire the original images with an additional fee.

In Section 6.1.1, the idea of the proposed lossless visible watermarking technique is described. Some properties of palette and graphics-interchange-format (GIF) images are introduced in Section 6.1.2. And in Section 6.1.3, we briefly describe the principle of the proposed method. In Section 6.2, the embedding and lossless recovery processes of the proposed method, the robustness against removal attacks, and the detailed algorithms are described. And in Section 6.4, some experimental results are shown. Finally, in Section 6.5, some discussions and a summary are made.

6.1.1 Idea of Proposed Robust Lossless Visible Watermarking Technique

In order to protect copyright of images, digital watermarking techniques, which include invisible watermarking and visible watermarking, are proposed for embedding a pre-defined watermark into an image. Traditionally, the processes of invisible watermark and visible watermark embedding usually result in image distortions. Even though the distortions are usually tiny, in some applications, such as medical and military, such distortions are unacceptable due to the high precision requirement in these applications. This intensifies the demand for lossless watermarking techniques, which are able to recover exactly the original images.

In addition to specific applications' requirements, lossless watermarking techniques can save image owners a lot of storage space. The owner can preserve only the stego-images instead of both the original images and their stego-versions, because by using lossless watermarking techniques they can obtain distortion-free original images from stego-images. Most of the lossless watermarking techniques focus on invisible watermarking [2]-[5], and few papers are found on lossy visible watermarking. A study is found in Hu *et al.* [9]. To the best of our knowledge, there is no paper on lossless visible watermarking of palette images so far.

Typical visible watermarking techniques were proposed for reducing the distortions of stego-images, so semi-transparent watermarks are embedded traditionally. However, we choose to embed opaque watermarks into cover images in this study. Although an opaque watermark results in more distortions visually than a semi-transparent watermark, but on the contrary, it has the advantages of better advertising effects and copyright declaration, and can be removed entirely by the proposed lossless watermarking technique when the image owner requires to get the original image.

6.1.2 Properties of Palette and GIF Images

A palette, in computer graphics, is a designated subset of the total range of colors supported by a computer graphic system. Each color in the palette is assigned a number, which is called an index number. This index determines the color of a pixel. A GIF image is an 8-bit palette image, and a portable–network-graphics (PNG) image can be an 8-bit-palette-based image, too.

The GIF image is adopted as the color palette image in our study. Before describing the method for the lossless watermarking technique for GIF images, some properties must be addressed. First, every GIF image contains at most 256 colors. Second, a color palette is used to store colors used in the image. Third, every pixel can be thought as an index number which is a reference to the color palette, so each pixel needs one byte of storage space. Finally, the GIF image uses a lossless compression algorithm, so there is no loss of any information when saving a GIF image.

6.1.3 Principle of Proposed Method

For the comprehensibility and the convenience to the following descriptions, some definitions are given first. The area in a cover image where a watermark is embedded will be called the *watermark area* and the remaining portion in the cover image will be called the *non-watermark one*. The pixels in the watermark with values 1 will be called *black watermark pixels*, denoted as W_b, and those with values 0 called *white ones* and denoted as W_w. The watermark is supposed to be embedded in the upper left corner of the image. The corresponding pixels of the cover image in the watermark area where W_b is supposed to be embedded will be called *black watermark* *embedded pixels* I_{Wb}, and the remaining portion in the watermark area will be called *white ones* I_{Ww}.

The basic idea of the proposed method is to replace the colors of I_{Wb} with other visually different colors and this replacement must be revertible. That is, to enable the embedded watermark looking visually different from the pixels adjacent to them and able to be removed losslessly with a right key. In order to make the embedded watermark look obviously different, we try to find the farthest color among the palette colors to replace with. In addition, we apply a two-level key protection to ensure the embedded visible watermark remaining visible when illegal users try to remove it without a right key.

Each visible watermark used for image copyright protection is assumed to be a binary image in this study. We first divide the cover image into a watermark area and a non-watermark area. We are only concerned about the watermark area. The watermark area is further divided into black watermark embedded pixels I_{Wb} which are going to be replaced with other colors, and white watermark embedded pixels I_{Ww} which will be kept unchanged. Then we count the occurrences of colors of I_{Ww}, and let the occurrences of color C_0 through C_{255} be denoted as O_0 through O_{255}. We sort the color palette by the occurrences O_i in a descending order. We call this sorted result a *sorted color palette* P_s. We propose an adjusted Euclidean distance by taking the concept of weighting into consideration, which is called a *weighted Euclidean distance*. A *weighted Euclidean distance* of color C_i is calculated by summing the products of the Euclidean distance between C_i and the other colors C_0 through C_{255} and the occurrences O_0 through O_{255}. The Euclidean distance between two colors C_1 and C_2 is defined as

$$\mu(c_1, c_2) = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \ . \tag{6.1}$$

And the weighted Euclidean distance of color C_i in the color palette is defined as

$$\mu(c_i) = \frac{\sum_{n=0}^{255} (Occurrence\ (c_n) \times \mu(c_i, c_n))}{\sum_{n=0}^{255} Occurrence\ (c_n)}. \qquad (6.2)$$

The function Occurrence(C_n) is the occurrence O_n of the color C_n counted only in colors of white watermark embedded pixels I_{Ww}. Taking this function as a weight helps us to find the farthest color among colors of I_{Ww} instead of the whole cover image.

Second, we find the farthest color from either the original color palette or the sorted color palette. What we want to choose is the color that has the largest weighted Euclidean distance. Using the weighted Euclidean distance helps us to find the farthest color among the colors of I_{Ww} instead of the whole cover image. We denote this farthest color by C_f, which is supposed to be the most different color from the colors of I_{Ww}. If the color of I_{Wb} is mostly replaced with C_f, the embedded watermark will look obviously different from its neighboring pixels, said I_{Ww}. After finding color C_f, we can set up a table beginning from the color in the palette which is closest to C_f to the color which is farthest to C_f. The result can be thought as a new color palette which has a different order from the original one. And we call this a *rearranged color palette* P_r.

Now we have a sorted color palette P_s and a rearranged color palette P_r. We can make a one-to-one mapping between the two palettes in their original order. That is, if the original color of the pixel ranks second in P_s, we map the color to the second color in P_r. Then we can replace all black watermark embedded pixels by this color mapping rule, and thus obtain a stego-image. As a consequence, we make most part of colors of I_{Wb} be replaced with a group of similar colors which are visually different from the colors of I_{Ww}.

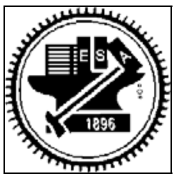
Finally, in order to prevent illegal users from removing the embedded watermark easily, we apply a two-level key protection when doing the color mapping. We add an offset generated by a secret key to the original order so that an illegal user cannot obtain the losslessly recovered image with a wrong offset generated by a wrong key. Moreover, there will be a noise-residual watermark on the recovered image. In addition to adding an offset, we randomize the position of the embedded watermark by the same key. The randomization results in an opaque embedded watermark, which has the advantage of better advertising effects and copyright declaration. The owner's copyright of the image can be protected further by this way.

6.2 Proposed Method for Copyright Protection of GIF Images

In Section 6.2.1, the visible watermark embedding process is introduced. And we describe the lossless recovery process of original images in Section 6.2.2. A noise-residual recovery process against removal attacks is presented in Section 6.2.3. Finally, detailed algorithms of embedding visible watermarks and lossless recovery of original images are described in Section 6.2.4.

6.2.1 Embedding of Removable Visible Watermarks by A Neighborhood-Dependent Watermarking Technique

In this section, the details of the proposed watermark embedding process are introduced. At first, we divide the watermark area into black watermark embedded pixels I_{Wb} and white ones I_{Ww}. Figure 6.1 shows an example of this.


(a)

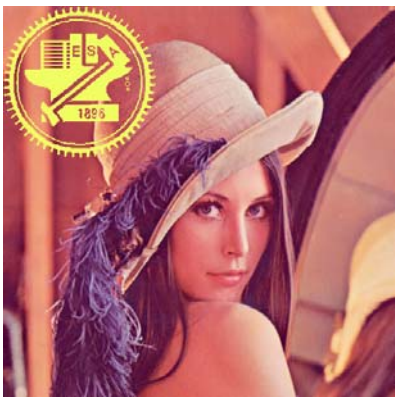
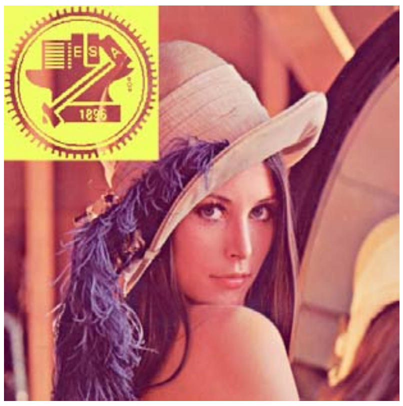
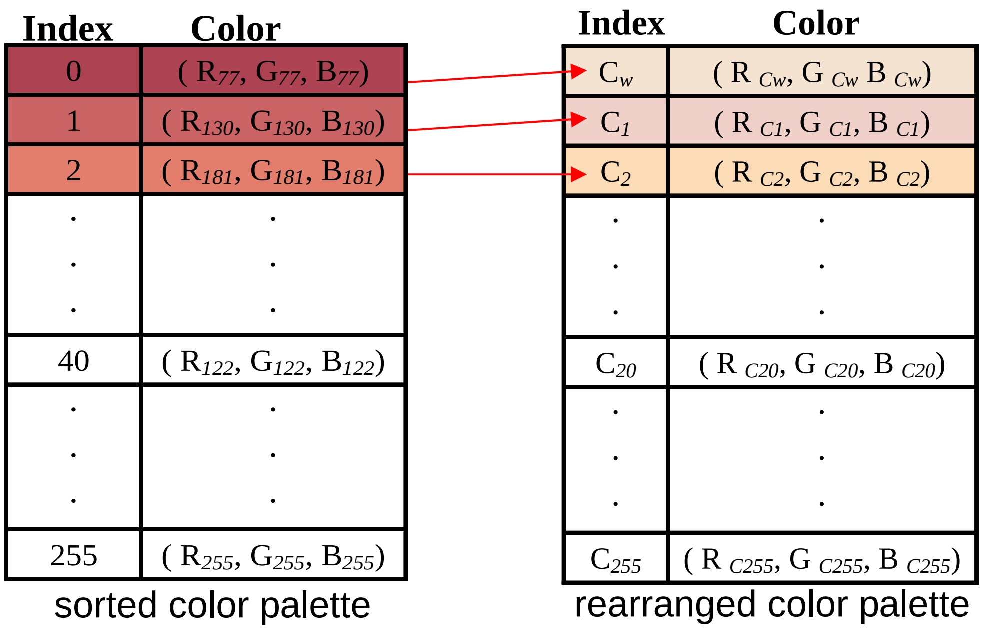


(b) (c)

Figure 6.1 An example of black watermark embedded pixels and white watermark embedded pixels. (a) Binary watermark. (b) Area of black watermark embedded pixels. (c) Area of white watermark embedded pixels.

And then we count the occurrences of the colors of white watermark embedded pixels I_{Ww}. A *sorted color palette* P_s can be set up by the occurrences in a descending order. The color which occurs with the highest frequency is ideally supposed to be replaced with a visually obviously different color. So we choose the farthest color from either the original color palette or the sorted one by picking a color C_f with a largest weighted Euclidean distance, and C_f will be the obviously different color to be replaced with. Once we have color C_f, we can find the closest color to C_f , which is called C_1, among the palette which we use to find color C_f by calculating the Euclidean distance between two involved colors. So we can set up a table from the closest color to the farthest color to C_f. We call this new table a *rearranged color* *palette* P_r. At this moment, we can embed a watermark into the cover image by replacing the colors of black watermark embedded pixels I_{Wb} with other obviously different colors according to P_s and P_r. If the original color ranks second in P_s, we map it to the second color in P_r. In other words, we make a one-to-one mapping from P_s to P_r in their original order Figure 6.2 shows an example of the one-to-one mapping from P_s to P_r by the original order. Finally, we apply a secret key in the process of doing the color mapping and randomizing the position of the embedded watermark for further protection.

A flowchart of the watermark embedding process is shown in Figure 6.3.



Index	Color
0	(R_{77}, G_{77}, B_{77})
1	(R_{130}, G_{130}, B_{130})
2	(R_{181}, G_{181}, B_{181})
. . .	. . .
40	(R_{122}, G_{122}, B_{122})
. . .	. . .
255	(R_{255}, G_{255}, B_{255})

sorted color palette

Index	Color
C_w	(R $_{Cw}$, G $_{Cw}$ B $_{Cw}$)
C_1	(R $_{C1}$, G $_{C1}$, B $_{C1}$)
C_2	(R $_{C2}$, G $_{C2}$, B $_{C2}$)
. . .	. . .
C_{20}	(R $_{C20}$, G $_{C20}$, B $_{C20}$)
. . .	. . .
C_{255}	(R $_{C255}$, G $_{C255}$, B $_{C255}$)

rearranged color palette

Figure 6.2 One-to-one mapping from sorted color palette to rearranged color palette by original order.

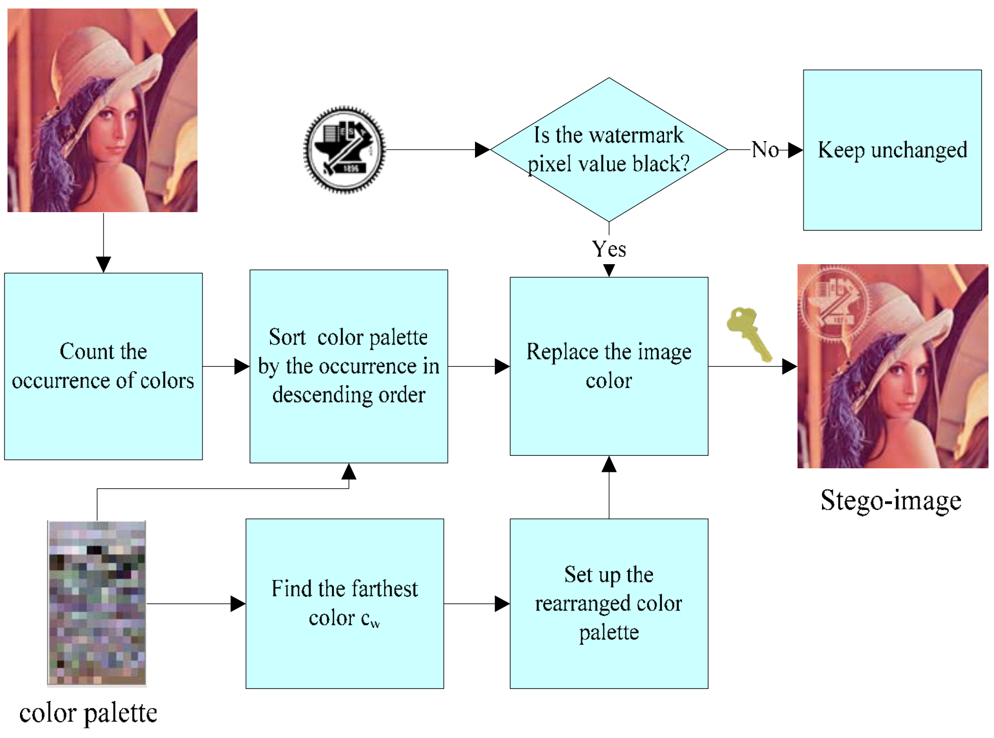

Is the watermark pixel value black?
No
Keep unchanged
Yes
Count the occurrence of colors
Sort color palette by the occurrence in descending order
Replace the image color
Stego-image
Find the farthest color c_w
Set up the rearranged color palette
color palette

Figure 6.3 A flowchart of proposed watermark embedding process.

6.2.2 Lossless Recovery of Original Images by Removal of Visible Watermarks

In this section, we describe how to recover the image losslessly and explain why we can achieve the goal.

In the watermark embedding process, we only replace the colors of I_{Wb}. Therefore, the colors of I_{Ww} keep unchanged. Moreover, the sorted color palette P_s and the rearranged color palette P_r are set up by only taking the colors of I_{Ww} into account. So in the recovery process, we can set up P_s and P_r which are the same as the two color palettes in the watermark embedding process. We let the corresponding area of I_{Wb} in the stego-image be denoted as S_{Wb} and the corresponding area of I_{Ww} in the stego-image as S_{Ww}. Then we find the order of each color of S_{Wb} in P_r, and recover it by the color in P_s with the right order and the right position. Certainly, if we have a right secret key, we will obtain the right order and the right position of every mapped color and consequently remove the watermark losslessly. A flowchart of the process of recovery of the original image is shown in Figure 6.4.

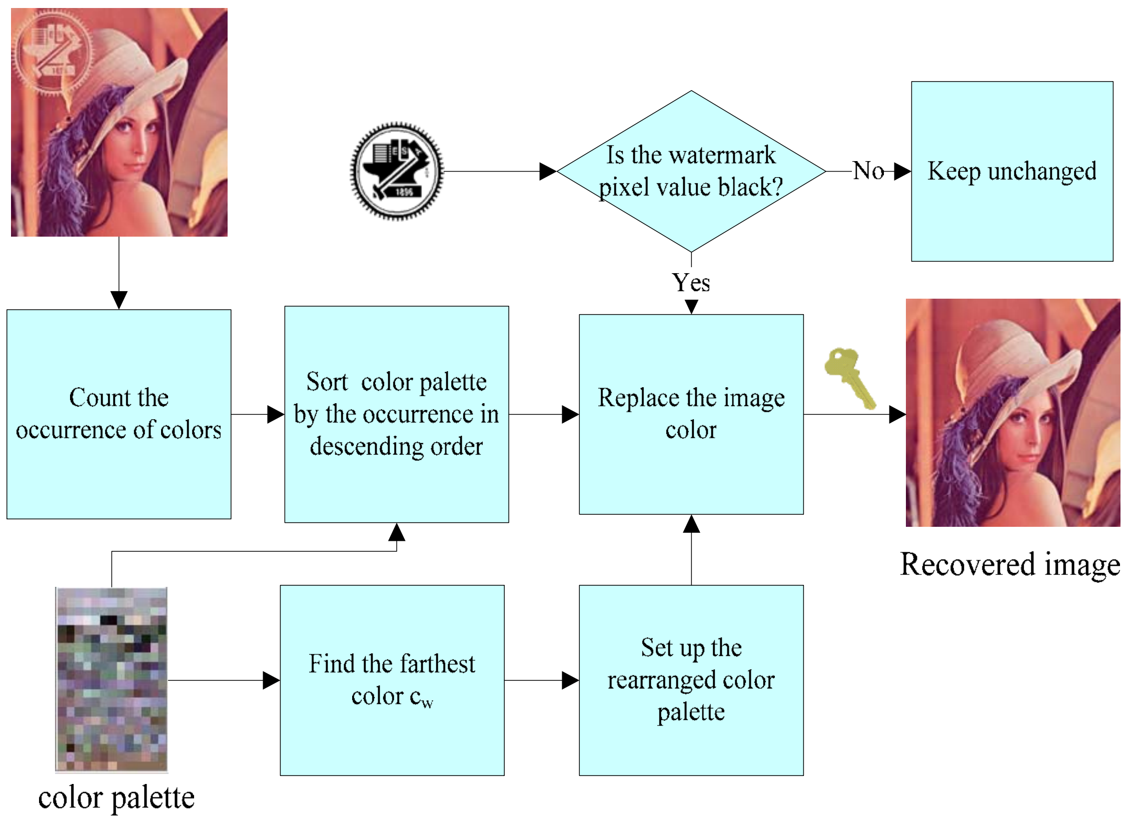

Is the watermark pixel value black?
No
Keep unchanged
Yes
Count the occurrence of colors
Sort color palette by the occurrence in descending order
Replace the image color
Recovered image
Find the farthest color c_w
Set up the rearranged color palette
color palette

Figure 6.4 A flowchart of proposed recovery process of stego-image.

6.2.3 Noise-Residual Recovery against Removal Attacks

In order to provide further protection of stego-images, we apply a secret key in the embedding process of doing the color mapping. We add an offset number between 0-20 into the original mapping order and obtain the mapped color from P_r, according to the new order. After embedding the watermark with this key protection, if an illegal user tries to remove the embedded watermark without a right key, a watermark-shaped noise will survive in the recovered image. The larger the range of the offset is, the worse the quality of the embedded watermark will be, but the more obviously the noise will survive in the recovered image by removing the embedded watermark with a wrong key. We make the lossless visible watermarking technique robust against removal attacks by this way.

Moreover, we randomize the position of the embedded watermark. The randomization results in an opaque watermark instead of a semi-transparent watermark. Although an opaque watermark visually distorts the image more than a semi-transparent watermark, it has the advantage of better advertising effects and copyright declaration. Figure 6.5 shows an example of the color mapping with an offset generated by a secret key.

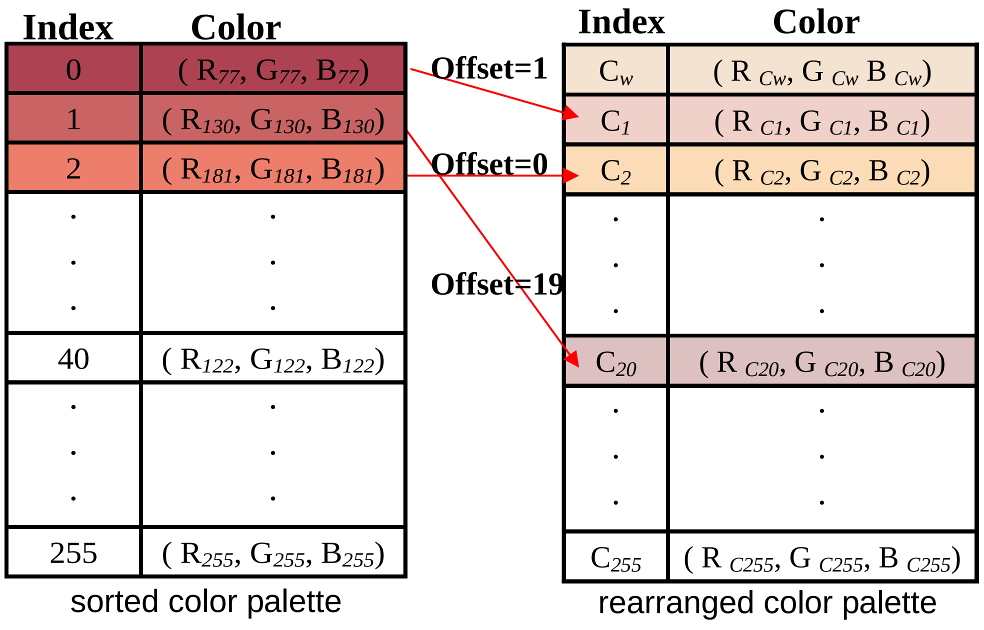


Figure 6.5 Color mapping with an offset generated by a secret key.

6.2.4 Detailed Algorithms

Detailed algorithms of the embedding process and the lossless recovery process are described below.

Algorithm 6.1: *Visible watermark embedding by a localized image-dependent method*.

Input: A cover image I, a watermark image W and a secret key K.

Output: A watermarked image S.

Steps:

1. Count the occurrences of colors of white watermark embedded pixels I_{Ww}.
2. Sort the color palette by the occurrences in a descending order and obtain a sorted color palette P_s.
3. Calculate the weighted Euclidean distance of every color in the color palette, and choose the color C_f which has largest weighted Euclidean distance to be the first color in a rearranged color palette P_r.
4. Calculate the Euclidean distance between C_f and the other colors in the color palette, and rearrange the color palette from the closest color to the farthest color compared to C_f. That is, set up the rearranged color palette P_r by the Euclidean distance compared to C_f in an ascending order.
5. Embed each watermark pixel t from the black watermark pixels W_b into the corresponding position of the original image in the following way.
   - 5.1 Find the color m of the original image in the corresponding position of the pixel t.
   - 5.2 Find the color order n of color m in P_s.
   - 5.3 Add an offset generated using K to the order n and find the new color m' by this new order n' in P_r.

5.4 Replace the color of the original image in the corresponding position of the pixel t with the new color m'.

6. Randomly pair the watermarked pixels together using K to form $|W_b|/2$ pairs of pixels and then swap the colors of the two pixels in each pair.

Algorithm 6.2: *Lossless recovery of original images.*

Input: A watermarked image S, the original watermark image W and a secret key K.

Output: A recovered image R.

Steps:

1. Count the occurrences of colors of white watermark embedded pixels S_{Ww}.
2. Sort the color palette by the occurrences in a descending order and obtain a sorted color palette P_s.
3. Calculate the weighted Euclidean distance of every color in the color palette, and choose the color C_f which has largest weighted Euclidean distance to be the first color in a rearranged color palette P_r.
4. Calculate the Euclidean distance between C_f and the other colors in the color palette, and rearrange the color palette from the closest color to the farthest color compared to C_f. That is, set up the rearranged color palette P_r by the Euclidean distance compared to C_f in an ascending order.
5. Pair the watermarked pixels together using K to arrive at the same $|W_b|/2$ pairs of pixels as that of watermark embedding and then swap the colors of the two pixels in each pair.
6. Remove each watermarked pixel t in the following way.

   6.1 Find the color s of the pixel t.

   6.2 Find the color order n of color s in P_r.

   6.3 Subtract an offset generated by K from the order n and find the new color m by this new order n' in P_s.

6.4 Replace the color of the pixel t with the new color m.

6.3 Experimental Results

Some experimental results of applying the proposed method are shown in this section. A Binary image of size 200×200 as shown in Figure 6.6 is used as the visible watermark. The cover images are all with size 512×512 and the watermarks are positioned in the upper left corner of the images. Figure 6.7(a) is the color image of "Lena" and (b) shows the resulting stego-image after embedding the visible watermark. Figure 6.7(c) shows the losslessly recovered image with (b), and (d) shows the lossy recovered images without a right key. And Figure 6.8 and Figure 6.9 are two other examples with "Painting" and "Jet," respectively. The corresponding Peak-Signal-to-Noise-Ratio (PSNR) values of (b) and (d) are shown in Table 6.1.

We measure the quality of a stego-image S by the *PSNR* value of the stego-image with respect to its cover image C. The mean square error MSE_S and the $PSNR_S$ for a stego-image after embedding a watermark W of size $M \times N$ are defined respectively to be

$$MSE_S = \frac{1}{M \times N}\sum_{x=1}^{M}\sum_{y=1}^{N}\left(S(x,y) - C(x,y)\right)^2 \quad \text{and} \quad PSNR_S = 10\log_{10}\left(\frac{255^2}{MSE_S}\right).$$

The *PSNR* values of the stego-images listed in the Table 6.1 are smaller than 15dB and show that the embedded watermarks are very different from its neighborhood of the original images. That is, the embedded watermarks look visually obviously different.

Table 6.1 PSNR values of images after embedding visible watermarks.

	Lena	Painting	Jet
stego-image	15.0	12.0	10.1

Besides, we measure the quality of a recovered image *R* by the *PSNR* value of the recovered image with respect to its original image *C*. The mean square error MSE_R and the $PSNR_R$ for a recovered image are defined respectively to be

$$MSE_R = \frac{1}{M \times N}\sum_{x=1}^{M}\sum_{y=1}^{N}\left(R(x,y) - C(x,y)\right)^2 \quad \text{and} \quad PSNR_R = 10\log_{10}\left(\frac{255^2}{MSE_R}\right).$$

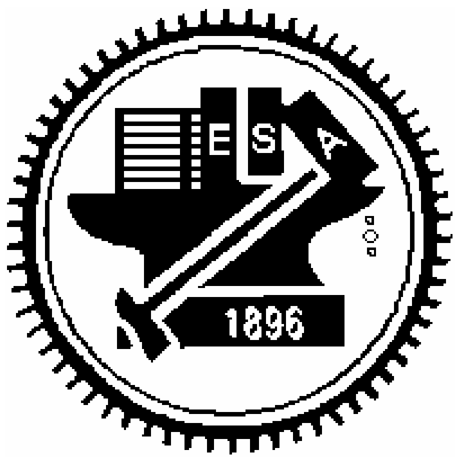
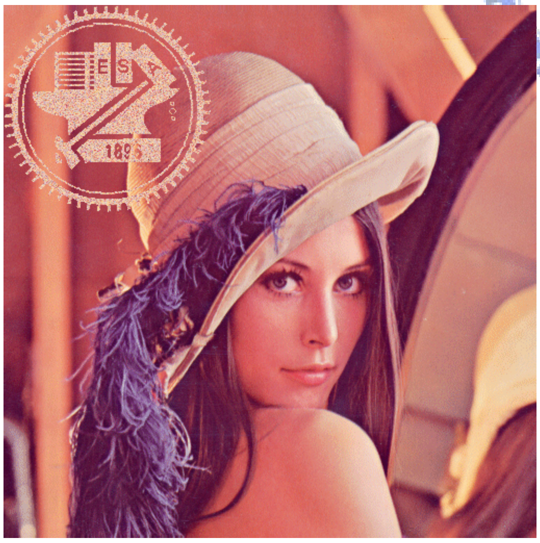
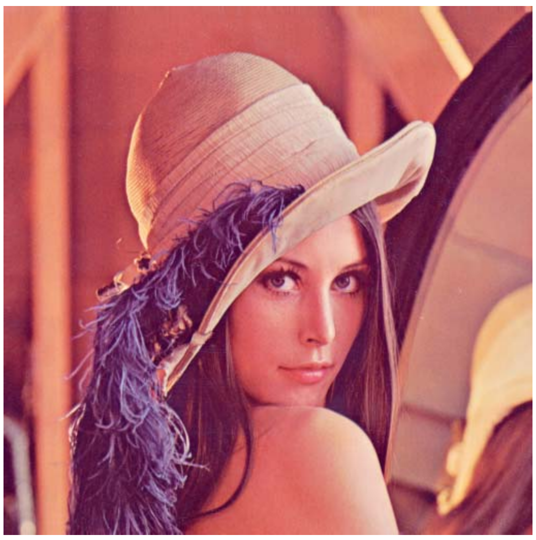


Figure 6.6 A binary watermark image of size 200×200.

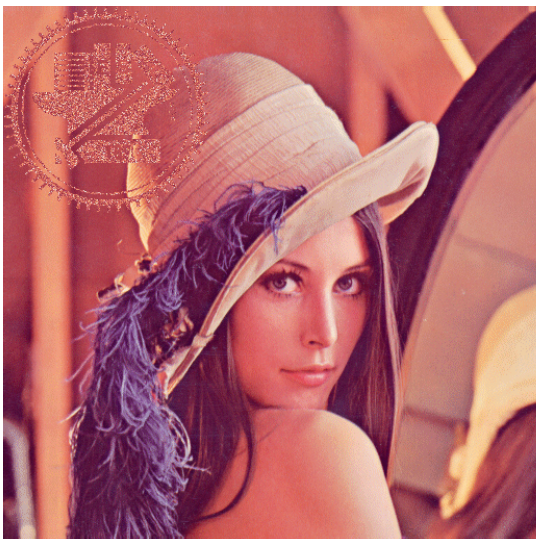
(a)



(b)



(c)



(d)

Figure 6.7 An example of results of applying proposed method. (a) A color image of "Lena". (b) Stego-image after embedding visible watermark of size 128×128. (c) Losslessly recovered image with (b). (d) Lossy recovered image without a right key.

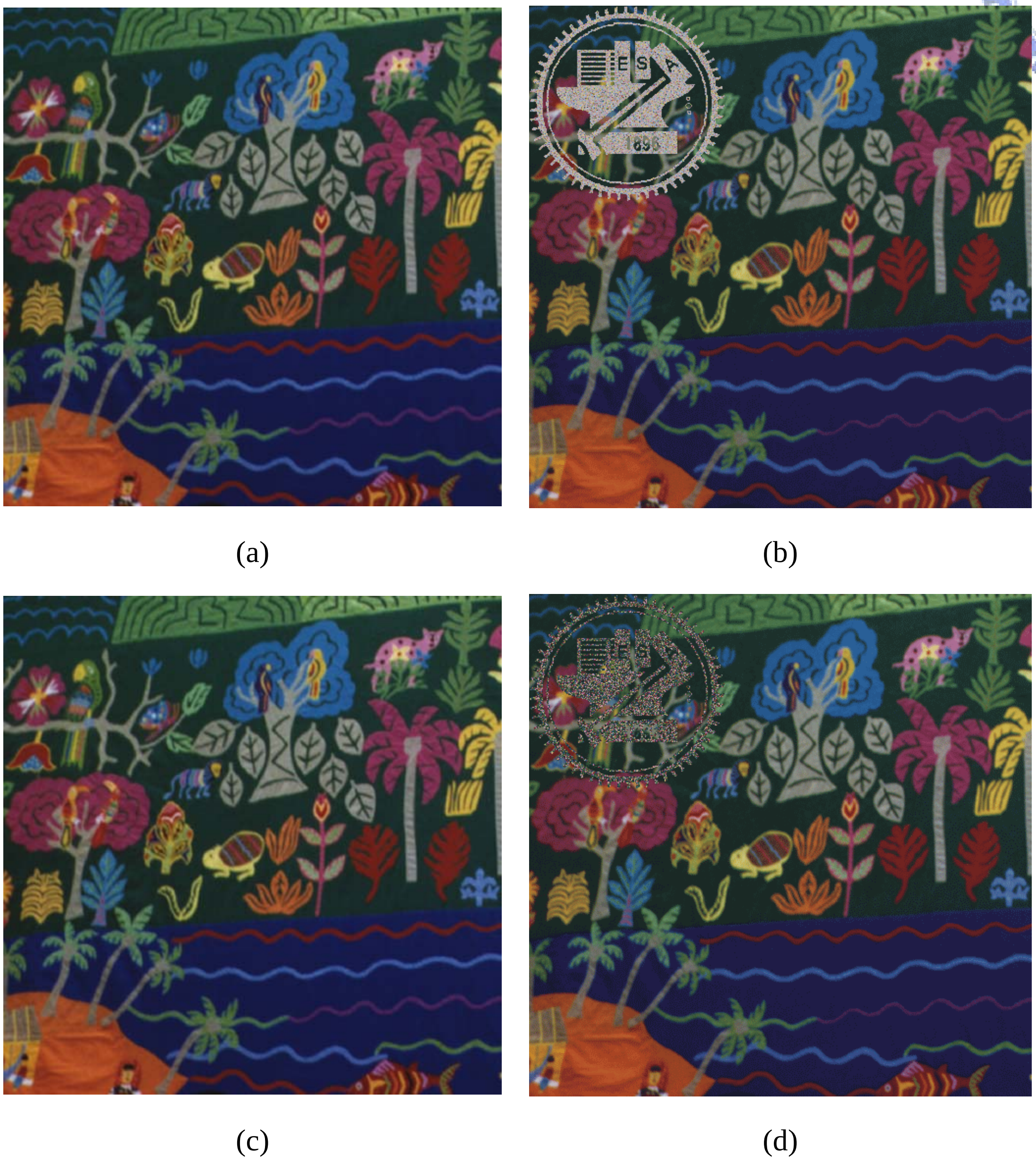
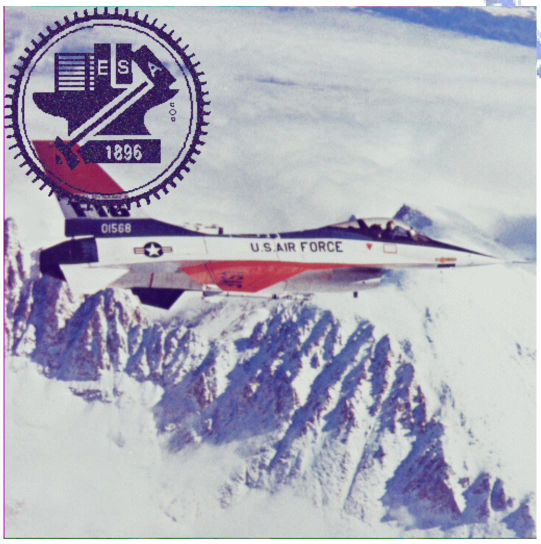
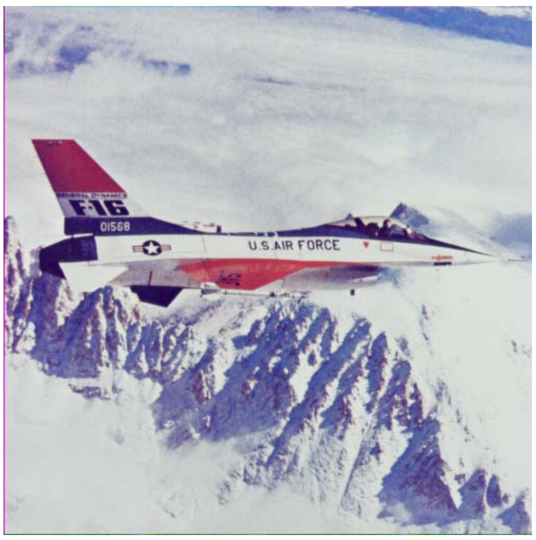
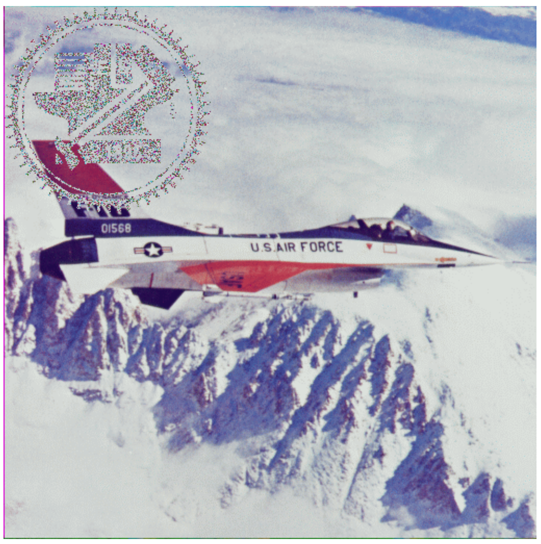


Figure 6.8 An example of results of applying proposed method. (a) A color image of "Painting". (b) Stego-image after embedding visible watermark of size 128×128. (c) Losslessly recovered image with (b). (d) Lossy recovered image without a right key.



(a) (b)





(c) (d)

Figure 6.9 An example of results of applying proposed method. (a) A color image of "Jet". (b) Stego-image after embedding visible watermark of size 256×256. (c) Losslessly recovered image with (b). (d) Lossy recovered image without a right key.

In the Table 6.2, the *PSNR* values of images recovered with a right key are all -1, which mean the MSE_R values are all zero. That is, the recovered images and the original images are exactly the same.

And the *PSNR* values of images recovered with a wrong key in are smaller than 20dB and show that the recovery results are still very different from the original ones due to the noise surviving in the watermarked area of recovered images by removing the embedded watermark with a wrong key.

Table 6.2 PSNR values of recovered images.

	Lena	Painting	Jet
Recovered with a right key	-1	-1	-1
Recovered with a wrong key	19.7	17.0	16.0

6.4 Discussions and Summary

In this chapter, we have presented a novel scheme to embed a removable visible watermark into GIF images. The proposed method can protect copyright of images. Two color palettes, a sorted color palette and a rearranged color palette, are set up. The sorted color palette is set up by sorting the counts of the occurrences of colors of white watermark embedded pixels in a descending order. And the first color of the rearranged color palette is assigned by the color with the highest weighted Euclidean distance. The remainder of the rearranged color palette is completed by calculating the Euclidean distances between the just assigned color and the other colors in the color palette and sorting the calculated Euclidean distances in an ascending order.

Eventually, we can embed the given watermark by mapping the colors of black watermark embedded pixels from the sorted color palette to the rearranged color palette.

In the recovery process of original images, we can set up the sorted color palette and the rearranged color palette which are the same as the two palettes in the embedding process because the only difference between the original image and the stego-image is the black watermark embedded pixels which does not influence the process of setting up the two color palettes. So we are able to remove the embedded watermark losslessly by mapping the colors of black watermark embedded pixels from the rearranged color palette to the sorted color palette.

Furthermore, we apply a secret key in two ways: one way enables the noise surviving in the recovered image without a right key, and the other makes the embedded watermark clearer and opaque. Under the assistance with the two-level key protection, we can protect the copyright of images further.

In the lossless recovery process, we need a stego-image and a watermark as inputs. However, it would be an interesting topic in the future to achieve lossless recovery results by restoring the images without the knowledge of the watermark.