DATA HIDING IN COLOR IMAGES BY COLOR REPLACEMENTS WITH REDUCTION OF IMAGE DISTORTION AND CHANGE NOTICEABILITY*

[1, 3]I-Shi Lee and I-Shi Lee(李義溪) [1, 2, ☿]Wen-Hsiang Tsai(蔡文祥)

[1]Dept. of Computer Science
National Chiao Tung University, Hsinchu, Taiwan 30010
[2] Dept. of Computer Science and Information Engineering
Asia University, Taichung, Taiwan 41354
[3]Dept. of Management Information
Technology and Science Institute of Northern Taiwan, Taipei, Taiwan

E-mails: gis87809@gmail.com & whtsai@cis.nctu.edu.tw

ABSTRACT

A new technique for data hiding in color images by color space partitioning and color encoding with reduction of image distortion and color change noticeability is proposed. The RGB color space is partitioned into non-overlapping, equal-sized color cubes. The colors in each cube are encoded to represent fixed-length message segments. Data hiding is accomplished by replacing the colors of the pixels with similar colors in the corresponding color cubes. And data extraction is a reverse process of data embedding. To reduce image distortion, color replacement at a pixel is conducted by choosing as the replacing color the one, which is closest to the pixel's color in the sense of Euclidean color distance. And to reduce the noticeability of the resulting color changes, color cubes are selected in such a way that the pixels of the cubes are as separated as possible in the cover image. Experimental results are also included, which show the effectiveness of the propose method in RGB images with high PSNR values and low visual differences in stego-images.

1. INTRODUCTION

Data hiding in images is a useful technique for many applications, such as copyright protection, covert communication, multimedia authentication, information sharing, etc. The image into which a message is hidden is called a *cover image*, and the result a *stego-image*. Many techniques for data hiding in images have been proposed in the past decade [1-3]. They may be categorized into two major approaches: the spatial-domain approach and the frequency-domain approach. In the former, secret data are directly embedded in the characteristics of the pixels of the cover image, and in the latter, the cover image is transformed first into frequency-domain coefficients, into which secret data are embedded. In general, the frequency-domain approach is more robust against attacks while the spatial-domain approach can hide more data. In this study, we adopt the spatial-domain approach and deal with the color image, like the BMP image.

The most common spatial-domain approach to data hiding is least-significant-bit (LSB) replacement, which embeds message data in the LSB's of image pixels' binary or grayscale values. Wang et al. [4] embedded data in the fifth LSB bit plane of a cover image, and employed an optimal substitution process based on a genetic algorithm and a local pixel adjustment method to lower the distortion in the stego-image. Chang et al. [5] used dynamic programming to obtain the optimal solution for the LSB substitution method. Chan and Cheng [6, 7] presented an optimal pixel adjustment process to improve the image quality of the stego-image acquired by Wang's schemes. Thien and Lin [8] proposed a method for hiding data digit by digit in images using a modulus function. The method is better than simple LSB substitution not only in eliminating false contours but in reducing image distortion. Lee and Chen [9] applied variable-sized LSB insertion to estimate the maximum embedding capacity by a human visual system (HVS) property, and to maintain image fidelity by removing false contours in smooth image regions. Wu and Tsai [10] presented a method based on the HVS by modifying quantization scales according to variation insensitivity from smooth to contrastive to improve stego-image quality. Lie and Chang [11] presented an adjusted LSB technique with the number of LSB's adaptive to the pixels of different grayscales. And Lee and Tsai [12] proposed a new method for data hiding in binary images based on block pattern coding and dynamic programming with distortion-minimizing capabilities. Tsai and Wang [13] proposed a data hiding technique for color images using a binary space partitioning tree, which partitions the RGB color space into voxels and embeds three message bits into each voxel.

In a conventional RGB image, there are 256^3, or approximately 16.8 million, colors which are the result from the use of 8 bits (one byte) for the color value of each of the red (R), green (G) and blue (B) channels. Human vision cannot distinguish such an enormous number of colors. Actually, even the 256 gray levels in a gray-scale image are still too many for visual discrimination. This kind of human vision weakness offers a good opportunity for data hiding by the technique of *color replacement*. That is, we may replace some pixels' colors in a cover image by other similar ones

☿To whom all correspondence should be sent.

without causing suspicion from observers. The proposed data hiding method is based on this principle.

More specifically, we propose in this study a new method for hiding data in RGB color images using *color space partitioning* and *color encoding*. The RGB color space is partitioned into non-overlapping, equal-sized color clusters, each being cubic in shape, called a *color cube*. The colors in each cube are used to represent fixed-length codes. Message data hiding is accomplished by replacing selected image pixels' colors with closest ones in color cubes to embed corresponding codes representing the message bits. And data extraction is a reverse process of data embedding. To reduce image distortion, each color cube is designed to include a number of color groups, with all colors in each group representing an identical code. The colors in each group are distributed as separately as possible in the cube, and color replacement at an image pixel is conducted by choosing as the replacing color the one in a group, which is *closest* to the pixel's color in the sense of Euclidean color distance. And to reduce the noticeability of the resulting color changes, we select adaptively for use in data embedding those cubes whose colors are *more scattered* in the cover image (that is, the pixels whose colors are in these cubes are more separated mutually in the cover image), so that the color changes on these pixels will arouse less notice from the observer. Experimental results show the effectiveness of the proposed method for large-volume data hiding with image distortion and change noticeability reduction.

In the remainder of this paper, the principle of the proposed data hiding method is described in Sections 2. The detailed algorithms of the data embedding and extraction processes are presented in Section 3. Some experimental results and discussions are given in Section 4, followed by conclusions in Section 5.

2. PRINCIPLE OF PROPOSED DATA HIDING METHOD

The basic idea of the proposed method for data hiding in RGB color images is to encode certain colors in the color space, and embed given message bits into selected scattered image pixels by replacing these pixels' colors by the encoded colors. And extraction of the message is a reverse process, consisting of finding image pixels with encoded colors and decoding these colors to get the embedded message bits.

Appropriate techniques must be devised for the above simple idea of data embedding and extraction to be carried out effectively. The concern of reducing image content distortion and color change noticeability should be taken into consideration in these techniques. Also, the common requirement of data recoverability in data extraction need be met. The techniques proposed in this study satisfy these aims and are described in the following.

A. Proposed technique for reduction of color change noticeability

It is unnecessary to use all of the huge number of colors in the color space for data embedding by color replacements. Instead, we partition them into *non-overlapping* cubic-shaped clusters, called *color cubes*, and find out those cubes *better* for use in data embedding. More specifically, we find out image pixels with their colors "falling" in each color cube, and check the *scattering* degree of these pixels. Presumably, image pixels located more separately in the cover image are more suitable for data embedding because the changes of their colors, appearing to be farther way from one another, will attract less notice from observers. On the contrary, color changes at less scattered pixels tend to create visual artifacts and arouse more suspicion. Based on this idea, we propose in this study the following scheme of reducing the noticeability caused by image pixels' color changes.

1. Partition the RGB color space into color cubes.
2. Collect the set of pixels in the cover image with their colors "falling" in each color cube, called the range set of the color cube.
3. Define the degree of *pixel scattering* of each color cube by a certain scatter measure of the pixels in the range set of the color cube.
4. Sort into a list the color cubes with *nonempty* range sets by their pixel scattering degrees, with the color cube with the largest scattering degree on the top of the list.
5. Sort further those color cubes with *equal* pixel scattering degrees by their range set sizes, meaning that color cubes with larger range sets will be used first for data embedding.
6. According to the length of the message to be hidden, select from the top of the color cube list a *sufficient* number of color cubes for use in data embedding.
7. Use the pixels of the range sets of the selected color cubes as the locations for data embedding by color replacements.

Let $S = \{P_1, P_2, \ldots, P_n\}$ denote the range set of a color cube C with n pixels. The scatter measure mentioned in Step 3 above for C, denoted as M, is defined as the mean of the Euclidean distances of all the pixel pairs in S, i.e., is defined as

$$M = \frac{\sum_{i,j} |P_i - P_j|}{n} \tag{1}$$

where the Euclidean distance $|P_i - P_j|$ between any two pixels Pi and Pj at image coordinates (u_i, v_i) and (u_j, v_j), respectively, is computed as $|P_i - P_j| = [(u_i - u_j)^2 + (v_i - v_j)^2]^{1/2}$. A larger value of M means higher pixel separateness of S in the cover image.

As an illustration of the range sets of color cubes, Fig. 1(a) shows a cover image and Fig. 1(b) is the range set of a color cube found in (a), shown as a binary image with each white dot indicating a pixel in the set. The range set may be seen to include pixels with some dark green colors.

B. Proposed technique for reduction of image content distortion

For convenience of data processing, the number of colors included in each color cube is taken to be a power of 2 in this study. If all the colors in a color cube, say with 2^m ones, are used for data embedding, each color may be used to represent m message bits. The embedding work of an m-bit message segment then is to replace the color of an image pixel in the range set of a color cube by the color of the 2^m ones in the color cube, which corresponds to the value of the m message bits.

However, to reduce the image distortion resulting from such color replacements, we propose in this study to allow *multiple* colors, instead of just a *single* one, to represent an identical message segment. For example, if we allow, say, 2^n colors as a group to represent a message segment, then whenever an image pixel's color is to be replaced by one in the color cube, there will be 2^n choices, and the one *closest* to the pixel's color may be taken as the replacing color, thus achieving the purpose of reducing image distortion due to the color replacement.

Consequently, each color cube as discussed above should be expanded to have $2^n \times 2^m = 2^{m+n}$ colors, instead of just 2^m ones, if embedding of m-bit message segments is still desired. And because of the property of having three color channels in an RGB image, $m+n$ must be a multiple of 3 for 2^{m+n} to be the cube of an integer M (i.e., M^3), meaning that each color cube has the side length of M. That is, it must be true that $m + n = 3k$ for some positive integer k such that $2^{m+n} = 2^{3k} = (2^k)^3 = (2^k) \times (2^k) \times (2^k) = M^3$ with $M = 2^k$. If not, then the color cluster will not form a cube; instead, it becomes a rectangular parallelepiped (also called a cuboid), which is less convenient to handle due to side asymmetry.

For example, if we take $m = 2$ and $n = 1$, then each color cube has $2^{2+1} = 8$ colors, divided into 4 groups with each group including two colors. One of such color cubes is shown in Fig. 2, in which the four color groups are $G_1 = \{(0, 0, 0), (1, 1, 1)\}$, $G_2 = \{(1, 0, 0), (0, 1, 1)\}$, $G_3 = \{(1, 1, 0), (0, 1, 1)\}$, $G_4 = \{(0, 1, 0), (1, 0, 1)\}$ and the two colors in each group are located diagonally in opposite directions, where each color is expressed as a 3-tuple (r, g, b) with r, g, and b being the values of the R, G, and B channels, respectively. Such color cubes are too small to be useful. The color cube adopted for use in the experiment of this study is taken to include $2^{m+n} = 2^{3+3} = 64$ colors with $m = 3$ and $n = 3$, i.e., with 8 groups of 8 colors. Therefore, for 8-bit R, G, and B color channels, there are totally $(256/4) \times (256/4) \times (256/4) = 64^3$ color cubes.

For convenience of discussions, we define a *base color* for each color cube as the one in the cube with the smallest of the summation of the r, g, and b values. By identifying color cubes with three indexes i, j, k for the three dimensions of R, G, and B, respectively, it is not difficult to figure out that the base color $(r_i^{\mathrm{b}}, g_j^{\mathrm{b}}, b_k^{\mathrm{b}})$ for the (i, j, k)-th color cube for $m = 3$ and $n = 3$ may be computed by

$$r_i^{\mathrm{b}} = 4i,\ g_j^{\mathrm{b}} = 4j,\ b_k^{\mathrm{b}} = 4k, \tag{2}$$

where $i, j, k = 0, 1, \ldots, 63$, and the values of the 64 colors in the cube may be computed by

$$\begin{aligned} r &= r_i^{\mathrm{b}},\ r_i^{\mathrm{b}}+1,\ r_i^{\mathrm{b}}+2,\ r_i^{\mathrm{b}}+3; \\ g &= g_j^{\mathrm{b}},\ g_j^{\mathrm{b}}+1,\ g_j^{\mathrm{b}}+2,\ g_j^{\mathrm{b}}+3; \\ b &= b_k^{\mathrm{b}},\ b_k^{\mathrm{b}}+1,\ b_k^{\mathrm{b}}+2,\ b_k^{\mathrm{b}}+3. \end{aligned} \tag{3}$$

For example, the (0, 0, 0)-th color cube with base color (0, 0, 0) is shown in Table 1. Simply adding the base color values $(r_i^{\mathrm{b}}, g_j^{\mathrm{b}}, b_k^{\mathrm{b}})$ respectively to the color channel values in the table, we can get the table for the (i, j, k)-th color cube.

According to the above idea, we propose the following scheme for reduction of image distortion.

1. Define each color cube to have 2^{m+n} colors, divided into 2^m groups with each group including 2^n colors, where $m + n = 3k$ for some positive integer k.
2. Assign the colors in the color cube into groups such that the colors in each group are distributed *evenly*, for the purpose of achieving more effectively the goal of reducing image distortion due to color replacements as discussed above.
3. Encode identically all the 2^n colors in each group into an m-bit message segment, i.e., represent the m message bits identically by any color in the group.
4. When an image pixel P in the range set of a color cube C is to be used for embedding an m-bit message segment H, find the group G in C whose colors represent the value of H.
5. Find the color c' in G which is closest to c in the sense of Euclidean color distance.
6. Replace c by c' to complete the data embedding work at pixel P.

In Step 5 above, the Euclidean color distance between the two colors $c = (r, g, b)$ and $c' = (r', g', b')$ are defined to be $|c - c'| = [(r - r')^2 + (g - g')^2 + (b - b')^2]^{1/2}$.

C. Proposed technique for extraction of embedded data

A merit of the previously-proposed technique of data hiding (including partitioning the color space into non-overlapping color cubes as well as replacing a pixel's color with another one, both in an identical color cube) is the resulting assurance of *data recoverability* in the data extraction stage. There are two reasons which guarantee this merit, as described in the following.

1. Although some original colors in the cover image have been replaced, each of the replacing colors is in the same color cube as that of the replaced one at an image pixel. This ensures that if we use the pixels' colors in the stego-image to find the range set of each color cube, as is done in the data extraction process, the result will be the same as that found in the data hiding process. This means that the pixels where data were hidden will not be missed in the data extraction process.
2. Only color cubes with more-scattered range sets are utilized for data embedding, and so if we select similarly color cubes with more-scattered range sets in the data extraction process, then the same set of color cubes will be found, from whose range sets we can extract exactly the previously-embedded message bits.

The proposed scheme for data extraction is described in the following.

1. Partition the color space in the same way as done in the data embedding process.
2. Collect the range set of each color cube from the pixels in the given stego-image.
3. Compute the scattering degree of the range set of each color cube.
4. Sort the color cubes into a list in the same way as done in the data embedding process described previously.
5. Select a sufficient number of color cubes from the top of the list according to the length of the embedded message.
6. Follow the color encoding rule used in data embedding to decode as a message segment the color of each pixel in the range set of each color cube selected in the last step.
7. Concatenate all the decoded message segments in order into a message as the extraction result.

In Step 5 above, to decide how many color cubes should be selected, the length of the message (in the unit of bit) should be known in advance. For this, we take the message length as part of the data to be hidden and append it to the message data as the prefix, in the form of a *fixed* number of bytes. If the value of the message length, expressed as a bit sequence, is shorter than the length of all the bytes allocated for it, then we pad sufficient leading 0's to it to fill up the bytes. In this way, the message length will be embedded first as a fixed number of bytes into the image, and in the data extraction process it can be extracted first as well from a fixed number of bytes hidden in the stego-image, from which the total number of remaining data bits can be decided, and the message bits extracted properly.

D. Even distribution of cube colors into groups for image distortion reduction

As mentioned previously, we assign the colors in the color cube into groups such that the colors in each group are distributed *evenly*. Consequently, a color in the group closest to an image pixel's color can be selected for color replacement, in order to reduce the resulting image distortion. Here we describe the technique we use for achieving such a goal of even distribution of colors in groups. First, it is not difficult to see that the desired distributions in the groups should be *symmetric* to each other. To accomplish this, we adopt the following steps, using the first color cube with base color (0, 0, 0) as an example for explanation of the detail. For other color

cubes, the corresponding steps are the same except the base color. Table 2 shows the details of the involved computation results in the steps.

1. Take the 64 color values of the color cube as Euclidean coordinates, and compute its centroid, which is (1.5, 1.5, 1.5).
2. Transform the Euclidean coordinates into new ones through a translation of (1.5, 1.5, 1.5).
3. Transform the new Euclidean coordinates (r, g, b) into 3D spherical coordinates (ρ, θ, ϕ) by the following formula:
   $\rho = (r^2 + g^2 + b^2)^{1/2}$, $\phi = \tan^{-1}(g/b)$, $\theta = \tan^{-1}[b/(r^2+g^2)^{1/2}]$
   where ρ is the distance from the origin to a point in the Euclidean space, θ is the zenith angle with respect to the R-axis, ϕ is the azimuth angle with respect to the B-axis, as shown in Fig. 3, and the function $\tan^{-1}$ has values in the range from -90^o to $+90^o$.
4. To facilitate the purpose of even distribution of group colors, modify the range of $\tan^{-1}$ such that the computed values of θ lie in the range $0^o \le \theta < 360^o$ with 0^o indicating the direction of the R-axis.
5. Use in order the values of ρ, ϕ, and θ to sort the 64 colors into a list.
6. Assign the 64 colors of the color cube evenly into the 8 groups using the list according to the following criteria to achieve the goal of even distribution of group colors:
   (1) each group has an equal number of colors which have a certain value of ρ;
   (2) the colors of each group have as many angles of θ as possible;
   (3) the 8 color groups, when seen as grid points, are symmetric to one another.
7. Regard all the 8 colors in each color group to be identical, and encode each group to represent one of the eight 3-bit segments 000 through 111, as mentioned previously.

In Step 6 above, to satisfy Criteria (2) and (3) we normalize the angle values of θ of all the grid points with respect to each of the angles of "8 selected symmetric points" and listed them for easier selection of appropriate colors into the groups. For the 64-color cubes, these 8 symmetric points may be selected to be the 8 corners of the cube, as done in our experiment. The result of color distribution for the first color cube with base color (0, 0, 0) is shown in Table 2. And an example of the color distribution result for group 3, which includes the corner of (0, 0, 0), is shown in Fig. 4. The assigned 8 colors in the group are (1, 2, 2), (3, 2, 2), (1, 0, 2), (2, 2, 3), (2, 3, 0), (0, 1, 0), (3, 0, 1), (0, 0, 0).

The above process is designed for color cubes with 64 colors. It is not difficult to modify the process to fit more general cases of color cubes with 2^{m+n} colors mentioned before.

Furthermore, as an example of data embedding at image pixels, let P be a pixel with color $c = (r, g, b) = (1, 3, 2)$ and assume that the 3-bit message segment we want to embed is 010. The color cube used is that described in Table 2 and the group of colors involved is the third shown in Fig. 4. The color in the group *closest* to c is $c' = (1, 2, 2)$ with a distance of 1 to c. Therefore, the color $c = (1, 3, 2)$ of P is replaced by $c' = (1, 2, 2)$ in the data embedding process.

As a deeper investigation of the effect of the above even distribution of group colors in a color cube, we tried to compute the value of the peak of the signal-to-noise ratio (PSNR) for the worst case of color replacements, which occurs when the colors of all image pixels are replaced with the most dissimilar colors in color cubes. For this, we have two cases. One is when the colors of each group in a color cube are *not* evenly distributed. Then, the largest Euclidean color distance resulting from a color replacement obviously will be $|(3, 3, 3) - (0, 0, 0)| = (3\times3^2)^{1/2} = \sqrt{27}$. The other case is when the even distribution is done as shown in Table 2. Then, according to a computer program written in this study which computes exhaustively the Euclidean color distances between every pair of colors in the color cube based on the groups of Table 3, the largest Euclidean color distance is $d = \sqrt{4}$.

Accordingly, for the 2nd case the maximum mean-square error (MSE) for the stego-image may be computed to be $MSE_{\max} = d^2/3 = 4/3$, and the corresponding worst PSNR value is $PSNR_{\min} = 10\times\log[255^2/MSE_{\max}] = 10\times\log[65025/(4/3)] \approx 46.88$ dB which is quite high. In contrast, the former case has $PSNR_{\min} = 10\times\log[65025/(27/3)] \approx 38.59$ dB which is lower. In short, the 2nd case, which is what we have implemented in this study, has less image distortion.

As a comparison with the LSB replacement method, the proposed method obviously performs better than the 3-LSB technique both in data hiding rate and in distortion reduction effect. It performs worse than the 1-LSB and 2-LSB techniques in data hiding but better in distortion reduction.

3. DETAILED ALGORITHMS OF PROPOSED DATA EMBEDDING AND EXTRACTION

We now describe the detailed algorithms for data embedding and extraction. We assume that the maximum length of given messages to be embedded is B bytes ($8B$ bits) long.

Algorithm 1. Data embedding process.

Input: a cover image I, a message G in the form of a bit string, and the color encoding tables (like Table 2) for color cubes with 64 colors defined by Eqs. (2) and (3).

Output: a stego-image I' with G embedded.

Steps:

A. ***Finding the range sets of the color cubes ---***

1. Find the range set S_i from the cover image I for each color cube C_i.
2. Compute the scattering degree M_i of each C_i by Eq. (1).
3. Sort all non-empty S_i into a list L according to their values of M_i with the top of the list corresponding to the largest M_i.

B. ***Creating extended message data ---***

4. Pad 0's, if necessary, to the front of the bit string representing the length of message G so that the resulting bit string, T, occupies B bytes.
5. Concatenate T and G in order, to form a third string T'.
6. Count the number of bits in T', append 0's to the end of T', if necessary, to make the total number N of bits a multiple of 3, call the resulting bit string an *extended message*, and denote it by G'.

C. ***Embedding of message data ---***

7. Regard all the pixels in each range set S_j in L in the raster-scan order as a sequence Q_j, and concatenate all sequences of Q_j in order into a longer one Q.
8. Embed sequentially every 3-bit segment H of G' into pixels in Q in order in the following way, until all bits of G' are exhausted:
   (1) take sequentially an unprocessed pixel P in Q with color c;
   (2) find out the color cube C whose range set includes P;
   (3) find out the color group p of C, whose corresponding code is equal to H;
   (4) find out the color c' in p which is *closest* to c in the

sense of Euclidean color distance;

(5) replace c of P by c' in the cover image.

The data extraction process is described as an algorithm in the following. We assume the embedded data in the given stego-image is the extended message G' mentioned in the previous algorithm, which includes the original message G preceded by the value of the length of G in the form of B bytes.

Algorithm 2. Data extraction process.

Input: a stego-image I', and the color encoding tables (like Table 2) for color cubes with 64 colors defined by Eqs. (2) and (3).

Output: the message G.

Steps:

A. Finding the range sets of the color cubes ---

1. Find the range set S_i from the stego image I' for each color cube C_i.
2. Compute the scattering degree M_i of each C_i by Eq. (1).
3. Sort all non-empty S_i into a list L according to their values of M_i with the top of the list corresponding to the largest M_i.

B. Extracting the length of the message

4. Regard all the pixels in each range set S_j in L in the raster-scan order as a sequence Q_j, and concatenate all sequences of Q_j in order into a longer one Q.
5. Extract B bytes of data from Q first to obtain the length N of the message G in the following way:
   (1) take sequentially an unprocessed pixel P in Q with color c';
   (2) find out the color cube C whose range set includes P;
   (3) find out the color group p of C, which includes c';
   (4) find out the 3-bit code corresponding to p;
   (5) repeat the above steps until the concatenation of all the found 3-bit codes in order, denoted as K, is *just more than* B bytes long;
   (6) take the first B bytes of K and convert it into an integer as the message length N, and the tail portion R in K as the leading bits of the message G.

C. Extracting the message data

6. Compute $N' = \lceil N/3 \rceil$ where $\lceil \cdot \rceil$ means the ceiling function.
7. Repeating the following steps N' times:
   (1) take sequentially an unprocessed pixel P in Q with color c';
   (2) find out the color cube C whose range set includes P;
   (3) find out the color group p of C, which includes c';
   (4) find out the 3-bit code of p;
8. Concatenate R extracted in Step 5 and all the codes extracted in Step 7 in order as a bit string, and take the first N bits of it as the desired message G.

4. EXPERIMENT RESULTS AND DISCUSSIONS

A series of experiments have been conducted in this study on BMP images. Some experimental results are shown in Figs. 5 through 8. Fig. 5 is a continuation of Fig. 1. Fig. 5(a) shows the stego-image resulting from embedding 22900 bytes of message data into the cover image shown in Fig. 1(a) which is of the size 256×256. And Fig. 5(b) shows the difference between Fig. 1(a) and Fig. 5(a) as a color image I'' (called a *difference image*), which is produced in the following way, assuming that (r, g, b) is a color in the cover image I, (r', g', b') the corresponding color in the stego-image I', and (r'', g'', b'') the computed difference color:

$$x'' = |x - x'| + 128 \quad \text{if } |x - x'| \neq 0;$$

$$= 255 \quad \text{if } |x - x'| = 0,$$

where $x = r$, g, or b. The concept behind the above computation is to set a difference value of 0 to be 255 and a non-zero one to be around 128. Consequently, an unprocessed pixel with three zero difference values will become a white pixel in the difference image I'', while a processed pixel will have a color (r'', g'', b'') with all the three color channel values around 128. As can be seen from Fig. 5(b), most of the pixels in the cover image have been utilized for data embedding, but the stego-image looks almost identical to the cover image of Fig. 1(a) due to the effectiveness of image distortion and change noticeability reduction. It can also be observed from Fig. 5(b) that the processed pixels are quite random in their locations, and more uniform regions, like those on the clothes, yield range sets with smaller scatter measures, as expected, which are not used for data embedding (seen as white-pixel clusters in the figure). The rate of processed pixels (called *processed pixel rate* in the sequel) is (22900×8) ÷ 3 ÷ (256×256) ≈ 0.932 and the PSNR value was computed to be 48.59 dB which is better than the worse-case value 46.88 dB, as it should be. Totally, 1628 color cubes have been utilized.

Fig. 6 shows another experimental result with a 256×256 cover image. The processed pixel rate is again 0.932, the computed PSNR value is 48.23 dB, and the number color cubes used is 5242. A similar phenomenon of leaving uniform regions unused for data embedding is observed (most on the flowers at the lower part of the cover image). For illustrations, we also include the range set of a color cube as Fig. 6(b). Two more examples of experimental results with 512×512 cover images are shown in Figs. 7 and 8. The message data embedded are 88200 bytes long, and the processed pixel rates are (88200×8) ÷ 3 ÷ (512×512) ≈ 0.897, for both cases. The PSNR values are 48.70 dB and 48.27 dB, respectively.

More statistics data about our experiments are shown in Table 3, in which images 4.1.03, 4.1.01, 4.2.04, 4.2.07 are those in Figs. 5 through 8, respectively. All the images come from the USC image database. From the table, we see that the PSNR values of all the stego-images are over 48 dB.

The experiments were conducted for color cubes with 64 colors and color groups of 8 colors. Color cubes and color groups of sizes other than those used in the experiments of this study may also be applied for various application needs. In general, larger-sized color cubes will lead to larger embedding capacity of each color replacement (that is, more bits are encoded by each replacing color) if the size of each color group is fixed. On the other hand, with the size of the color cube being fixed, larger-sized color groups, though reducing more distortion caused by color replacements, will lead to less embedding capability (that is, less bits are encoded by each color group). The original cover image is not needed in data recovery, so the proposed method is a blind scheme. The PSNR values of the stego-images constructed in the experiments are high, showing that the aim of image distortion reduction carried out by the use of color groups is accomplished. The stego-images look almost identical to the cover images, showing that another aim of reducing color change noticeability is also reached. Furthermore, secret keys may be used to randomize the message data before they are embedded into the cover image or/and randomize the sequence of pixels (sequence Q in Algorithms 1 and 2) into which the data are embedded, in order to enhance data security. Illegal recovery of the embedded data will so obtain just a sequence of noise. The proposed method is thus appropriate for uses in steganographic applications.

5. CONCLUSIONS

A novel method for hiding large-volume message data in RGB images has been proposed. The method is based on the idea of changing selected image pixels' colors by similar ones which encode the message bits. The replacing colors come from some selected color cubes in the color space, and the image pixels come from the range sets of the color cubes. Data recoverability is ensured by the use of color cubes and range sets. The color cubes are selected in such a way that the pixels in their range sets are as separated as possible. This reduces the noticeability caused by the color changes. Each replacing color comes from the choice of an optimal one from a group of evenly distributed colors in a color cube. This reduces the resulting image distortion due to the color replacements. Experimental results show the feasibility of the proposed method for large-volume data hiding as well as the effectiveness of reducing image distortion and change noticeability. The method is a blind data hiding technique; the original cover image is not required in the data extraction process. Future researches may be directed to dynamic uses of variable-sized color cubes, random distributions of groups' colors in color cubes, uses of the proposed method for various applications, etc.

REFERENCES

[1] S. Katzenbeisser and F. A. P. Petitolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston, U. S. A., 2000.

[2] M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia data-embedding and watermarking technologies," *Proceedings of the IEEE*, vol. 86, pp. 1064-1088, 1998.

[3] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM System Journal*, Vol. 35, No. 3 & 4, Feb. 1996.

[4] R. Z. Wang, C. F. Lin, and J. C. Lin, "Hiding data in Images by optimal moderately-significant-bit replacement," *IEEE Electronics Letters*, vol. 36, no. 25, pp. 2069-2070, Dec. 2000.

[5] C. C. Chang, J. Y. Hsiao, and C. S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy," *Pattern Recognition*, vol. 36, pp. 1583-1595, 2003.

[6] C. K. Chan, and L. M. Cheng, "Improved hiding data in images by optimal moderately-significant-bit replacement," *IEEE Electronics Letters*, vol. 37, no. 16, pp. 1017-1018, August 2001.

[7] C. K. Chan, and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, vol. 37, pp. 469-474, 2004.

[8] C. C. Thien and J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function," *Pattern Recognition*, vol. 36, pp. 2875-2881, 2003.

[9] Y. K. Lee and L. H. Chen, "High capacity Image steganographic model," *IEE Proceedings on Vision, Image Signal Process*, vol. 147 no. 3, June 2000.

[10] D. C. Wu and W. H. Tsai, "Spatial-domain image hiding using an image differencing," *IEE Proceedings-Vision, Image, and Signal Processing*, vol. 147, no. 1, pp. 29-37, 2000.

[11] W. N. Lie, and L. C. Chang, "Date hiding in images with adaptive numbers of least significant bits based on the human visual system," *Proceedings of IEEE International Conference on Image Processing*, Taipei, Taiwan, Republic of China, vol. 1, pp. 286-290, October 1999.

[12] I. S. *Lee* and W. H. Tsai, "Data hiding in binary images with distortion-minimizing capabilities by optimal block pattern coding and dynamic programming techniques," *IEICE Transactions on Information and Systems,* Vol. E90-D, No. 8, pp. 1142-1150, 2007.

[13] Y. Y. Tsai and C. M. Wang, "A novel data hiding scheme for color images using a BSP tree," *Journal of Systems and Software*, Vol. 80, pp. 429–437, 2007.

ACKNOWLEDGEMENT

This work was supported partially by the NSC Advanced Technologies and Applications for Next Generation Information Networks (II) – Sub-project 5: Network Security, Project No. NSC-96-2752-E-009-006-PAE. And partially by the NSC project NSC96-2422-H-009-001.

Table 1. The colors in the (0, 0, 0)-th color cube with base color (*r*, *g*, *b*) = (0, 0, 0).

No.	Color	No.	Color	No.	Color
1	(0, 0, 0)	23	(1, 1, 2)	45	(2, 3, 0)
2	(0, 0, 1)	24	(1, 1, 3)	46	(2, 3, 1)
3	(0, 0, 2)	25	(1, 2, 0)	47	(2, 3, 2)
4	(0, 0, 3)	26	(1, 2, 1)	48	(2, 3, 3)
5	(0, 1, 0)	27	(1, 2, 2)	49	(3, 0, 0)
6	(0, 1, 1)	28	(1, 2, 3)	50	(3, 0, 1)
7	(0, 1, 2)	29	(1, 3, 0)	51	(3, 0, 2)
8	(0, 1, 3)	30	(1, 3, 1)	52	(3, 0, 3)
9	(0, 2, 0)	31	(1, 3, 2)	53	(3, 1, 0)
10	(0, 2, 1)	32	(1, 3, 3)	54	(3, 1, 1)
11	(0, 2, 2)	33	(2, 0, 0)	55	(3, 1, 2)
12	(0, 2, 3)	34	(2, 0, 1)	56	(3, 1, 3)
13	(0, 3, 0)	35	(2, 0, 2)	57	(3, 2, 0)
14	(0, 3, 1)	36	(2, 0, 3)	58	(3, 2, 1)
15	(0, 3, 2)	37	(2, 1, 0)	59	(3, 2, 2)
16	(0, 3, 3)	38	(2, 1, 1)	60	(3, 2, 3)
17	(1, 0, 0)	39	(2, 1, 2)	61	(3, 3, 0)
18	(1, 0, 1)	40	(2, 1, 3)	62	(3, 3, 1)
19	(1, 0, 2)	41	(2, 2, 0)	63	(3, 3, 2)
20	(1, 0, 3)	42	(2, 2, 1)	64	(3, 3, 3)
21	(1, 1, 0)	43	(2, 2, 2)		
22	(1, 1, 1)	44	(2, 2, 3)		

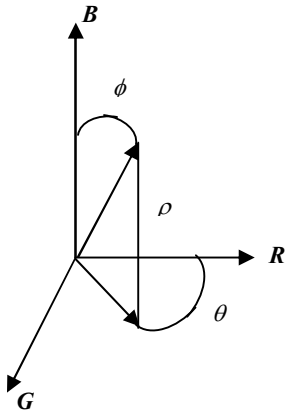
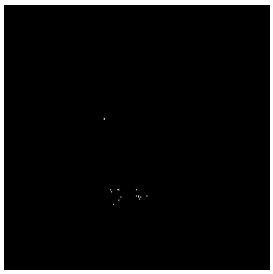


Fig. 3. Illustration of a 3D spherical coordinate system for use in even color distribution.

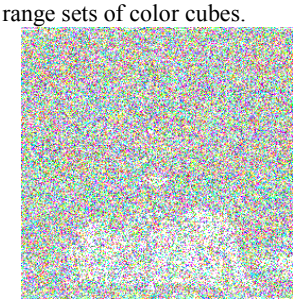
(a) Cover image.



(b) Range set of a color cube.

Fig. 1. An illustration of range sets of color cubes.

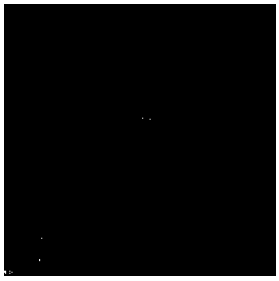
(a) Stego-image.



(b) Difference image.

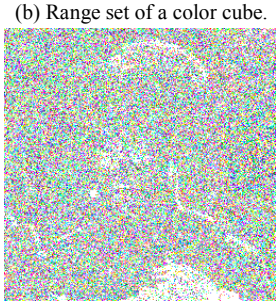
Fig. 5. An experimental result of data embedding applied to Fig. 1(a) with a 256×256 cover image and a 22900-byte message.

(a) Cover image.



(b) Range set of a color cube.

(c) Stego-image.



(d) Difference image.

Fig. 6. A second experimental result with a 256×256 cover image and 22900-byte message.

(a) Cover image.

(b) Stego-image.

(c) Difference image.

Fig. 8. A fourth experimental result of data embedding with a 512×512 cover image and a 88200-byte message.

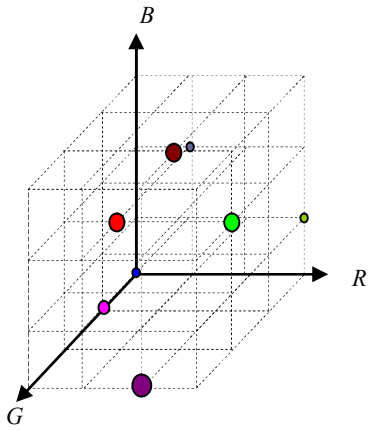
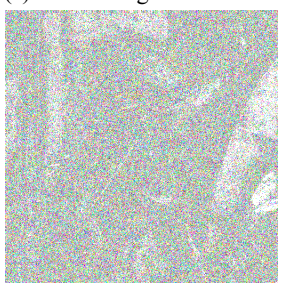


Fig. 4. An example of color distribution in a color cube .--- the 8 colors in group 3.

(a) Cover image.

(b) Stego-image.



(c) Difference image.

Fig. 7. A third experimental result of data embedding with a 512×512 cover image and a 88200-byte message.

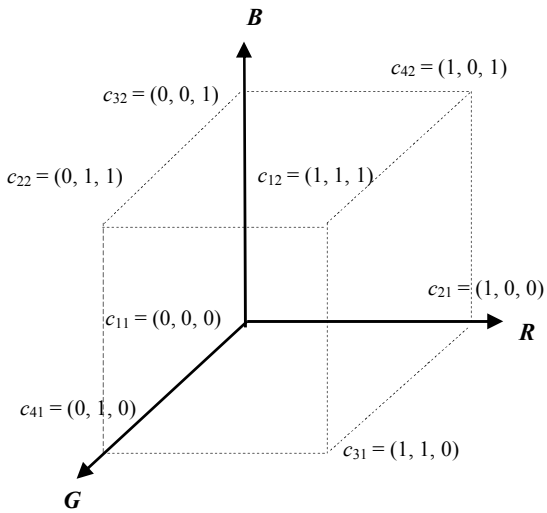


Fig. 2. A color cube with 8 colors divided into four groups with base color (0, 0, 0).

Table 2. Color encoding table for the (0, 0, 0)-th color cube with base color (0, 0, 0).

r	g	b	ρ	ϕ	θ	group	code
2	1	2	0.9	35	315	1	000
2	3	2	1.7	18	72		
0	1	2	1.7	18	198		
1	1	3	1.7	65	225		
3	2	0	2.2	-43	18		
1	0	0	2.2	-43	252		
0	3	1	2.2	-13	135		
3	3	0	2.6	-35	45		
2	2	2	0.9	35	45	2	001
0	2	2	1.7	18	162		
2	0	2	1.7	18	288		
2	1	3	1.7	65	315		
1	3	0	2.2	-43	108		
3	1	0	2.2	-43	342		
0	0	1	2.2	-13	225		
0	3	0	2.6	-35	135		
1	2	2	0.9	35	135	3	010
3	2	2	1.7	18	18		
1	0	2	1.7	18	252		
2	2	3	1.7	65	45		
2	3	0	2.2	-43	72		
0	1	0	2.2	-43	198		
3	0	1	2.2	-13	315		
0	0	0	2.6	-35	225		
1	1	2	0.9	35	225	4	011
1	3	2	1.7	18	108		
3	1	2	1.7	18	342		
1	2	3	1.7	65	135		
0	2	0	2.2	-43	162		
2	0	0	2.2	-43	288		
3	3	1	2.2	-13	45		
3	0	0	2.6	-35	315		
2	1	1	0.9	-35	315	5	100
1	1	0	1.7	-65	225		
2	3	1	1.7	-18	72		
0	1	1	1.7	-18	198		
0	3	2	2.2	13	135		
3	2	3	2.2	43	18		
1	0	3	2.2	43	252		
3	3	3	2.6	35	45		
2	2	1	0.9	-35	45	6	101
2	1	0	1.7	-65	315		
0	2	1	1.7	-18	162		
2	0	1	1.7	-18	288		
0	0	2	2.2	13	225		
1	3	3	2.2	43	108		
3	1	3	2.2	43	342		
0	3	3	2.6	35	135		
1	2	1	0.9	-35	135	7	110
2	2	0	1.7	-65	45		
3	2	1	1.7	-18	18		
1	0	1	1.7	-18	252		
3	0	2	2.2	13	315		
2	3	3	2.2	43	72		
0	1	3	2.2	43	198		
0	0	3	2.6	35	225		
1	1	1	0.9	-35	225	8	111
1	2	0	1.7	-65	135		
1	3	1	1.7	-18	108		
3	1	1	1.7	-18	342		
3	3	2	2.2	13	45		
0	2	3	2.2	43	162		
2	0	3	2.2	43	288		
3	0	3	2.6	35	315		

Table 3 Statistics of experimental results.

No.	Image	Size of message data (bytes)	Process-ed pixel rate	No. of used color cubes	PSNR (dB)
1	4.1.01(256×256)	22900	0.932	5242	48.23
2	4.1.02(256×256)	22900	0.932	3329	48.49
3	4.1.03(256×256)	22900	0.932	1628	48.59
4	4.1.05(256×256)	22900	0.932	3840	48.68
5	4.2.01(512×512)	88200	0.879	5514	48.36
6	4.2.02(512×512)	88200	0.879	5446	49.19
7	4.2.04(512×512)	88200	0.879	9908	48.70
8	4.2.05(512×512)	88200	0.879	6626	48.61
9	4.2.06(512×512)	88200	0.879	17093	48.60
10	4.2.07(512×512)	88200	0.879	17110	48.27
11	House(512×512)	88200	0.879	16048	48.68