ABSTRACT

Information security has recently become important in a number of application areas. Digital audio, video and pictures are increasingly furnished with distinguishing but imperceptible marks. Palette images are widely used in multimedia and Internet applications. In this project, a new method for data hiding in palette images with security protection by color ordering and mapping is suggested. First, image pixels are classified as data embeddable or non embeddable, and only the former ones are used to embed secret data. The proposed idea of data hiding is based on the use of a new type of color-ordering relationship, from which a color-mapping function is defined with binary values as output. When a secret data bit is to embed, a data embeddable pixel is selected; its color is adjusted to make the output of color-mapping function equal to the secret bit value. The embedded secret data can be extracted correctly and quickly from the resulting stego-image by merely inspecting the outputs of the color-mapping function.

This paper contains modified algorithm [1] in which parameters are randomized. It contains introduction, background, implementation, testing and result analysis, conclusion and future enhancement.

KEYWORDS
Palette images, Stenography, Secret data embedding

1. Introduction:

Fridrich and Du [6] proposed a method to embed data in palette images by first assigning a specific parity bit (0 or 1) to each color in the palette. Then adjust the pixel index values in such a way that the parities of the new index values are equal to the message bits to be embedded. They proposed an optimal parity-assignment algorithm so that the resulting data-embedding process guarantees that an index is always replaced by the index of the closest color. Their adaptive method can be employed to conceal a moderate amount of data and has the least modification of pixel values. However, replacing a color with its closest one may not always be the best choice. Theoretically, minimization of the color difference after data embedding will result in the least distortion in the image. Practically, this might not be adequate for palette images, especially for those with low color depth.

Palette based images, or simply palette images, are popular in multimedia and Internet applications. Each palette image composed of a color palette and a set of color indexes. The color palette is a set of entries of representative colors in the image. The color indexes are some pointers to those palette entries that specify the red-green-blue (RGB) colors in the images. Use of this type of palette image format has the effect of image compression. It helps saving storage space and reducing transmission time.

In most small web graphics which are saved in GIF format should never exceed 10KB. Ex: Graphics Interchange Format (GIF). There are 256 possible permutations of the 256 entries of a color palette. So, at most \( \log_2(256!) \) bits can be embedded into a GIF image.

Various palette images are available for secret data embedding, so one may choose images with larger numbers of colors as cover images. This method also can be applied to palette images with small numbers of colors except that the embedding capacity of such images will be smaller [6]. In case that a large amount of data is to be hidden, one way out is
to use multiple images instead of just one for data hiding. The array of multiple GIF images is called Animated Gifs. They are created by piecing together separate frames and assigning delay between the displays of the different frames. In this manner, images can appear to be animated, or in a state of change.

GIF image consists of two components:

As in .gif file only 256 colors are available, the color palette is also having 256 color entries. The size of the index file is equal to the size of an image.

2. Background
Steganography techniques are classified as adaptive steganography techniques and non-adaptive steganography techniques. In this proposed method adaptive steganography is used.

2.1 Selection of Optimal Color

![Figure 1: Replacement of color](image)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

Figure 1 illustrates an example of the observation about this phenomenon [1]. For simplicity, only three colors $c_1$, $c_2$, and $c_3$ with RGB values $(0, 174, 239)$, $(57, 181, 74)$, and $(236, 0, 140)$ are contained in the sample image.

1. Figure 1 (a) shows the original image. The pixel to be adjusted is located near the center of the image and is marked by a bounding box. It can be noted that the original color of the pixel is $c_2$.
2. Because the closest color of $c_2$ is $c_1$, $c_2$ is replaced by $c_1$ to minimize the color change of the pixel, as shown in figure 1(b). This results in a spark in the image.

3. Figure 1(c) shows the result of replacing $c_2$ with the neighboring color $c_3$. The image contains no visual artifact. As a result, in the development of data hiding in palette images, not only the color characteristics, but also the spatial properties of the image content should be considered.

This method is for steganographic applications [2]. Secret information hidden in a stego-image is fragile to image manipulation. Also, such applications require that the secret information hidden in a stego-image is visually and statistically undetectable, and the hidden secret cannot be read even when its existence is detected. To achieve these two goals, the proposed scheme is based on the use of a certain color-ordering relationship. Such an ordering relationship is designed according to some color quality. The ordering relationship is used to choose an optimal set of colors to replace the original ones to achieve better hiding effects based on the idea of minimizing color distortion. Besides, the proposed method also considers the necessity of reducing local image content changes as mentioned above, yielding secret embedding results with a good compromise between the resulting visual quality and data-hiding capacity.

2.2 Color-Mapping using color-ordering relationship

The binary color-mapping function [1] takes as input the color values of a group of image pixels. It yields as output a binary value “0” or “1” in accordance with the color relationship among the pixel’s colors. The basic idea of data embedding is to modify the colors of some image pixels. Color is modified in such a way that the corresponding binary outputs of the color-mapping function with the colors of the resulting image pixels as input are equal to the data bits to be hidden.

**Color-ordering relationship:**

Let $c_1$ and $c_2$ be two colors with RGB values $(r_1, g_1, b_1)$ and $(r_2, g_2, b_2)$, respectively. The color-ordering relationship is based on the luminance values $v_1$ and $v_2$ of $c_1$ and $c_2$, where $v_1$ and $v_2$ are computed as follows:

$$v_1 = 0.3 \times r_1 + 0.59 \times g_1 + 0.11 \times b_1$$
$$v_2 = 0.3 \times r_2 + 0.59 \times g_2 + 0.11 \times b_2 \quad \text{--- (1)}$$
It is possible that $c_1$ and $c_2$ are different, but $v_1$ and $v_2$ are equal to each other. In this situation, the color-ordering relationship is defined by comparing the RGB values of $c_1$ and $c_2$ further to make the relationship unique. Three possible color orders between $c_1$ and $c_2$ are defined as follows, which we say to form a color-ordering relationship $R_{cm}$:

$R_{cm}: c_1 > c_2$, if $(v_1 = v_2)$ or $(v_1 = v_2$ and $r_1 > r_2$) or $(v_1 = v_2$ and $r_1 = r_2$ and $g_1 > g_2$)

$c_1 = c_2$, if $(r_1 = r_2$ and $g_1 = g_2$ and $b_1 = b_2$)

$c_1 < c_2$, otherwise

(2)

**Color-mapping function:**

In this method [1], an input cover image is processed in a raster-scanning manner. Given a pixel $X$ in the cover image, we define its precedent neighbors to be those four neighboring pixels, among the eight neighboring ones in a 3 x 3 neighborhood of $X$, which are visited in sequence before the other four during the line-by-line raster scanning. More specifically, if $X$ is to be located at coordinates $(i,j)$ in the input image, then its precedent neighbors are the four pixels located at coordinates $(i-1,j-1),(i,j-1),(i+1,j-1)$ and $(i-1,j)$.

Figure 2 shows a pixel $X$ and its four precedent neighbors in the raster-scanning sequence.

Figure 2: Pixel X and its four precedent neighbors.

Color difference can be calculated by three different ways:

1. The color difference $|c_1 - c_2|$ between two colors $c_1$ and $c_2$ is the Euclidean distance between the RGB values $(r_1, g_1, b_1)$ and $(r_2, g_2, b_2)$ of $c_1$ and $c_2$, respectively, or more specifically, as $|c_1 - c_2| = [(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2]^{1/2}$.
2. The color difference between two pixels $X_1$ and $X_2$ with colors $c_1$ and $c_2$, respectively, as the color difference between $c_1$ and $c_2$.
3. The maximum color difference between a given pixel $X$ and its four precedent neighbors $X_1$ - $X_4$ as the maximum of the four color differences between $X$ and $X_1$ - $X_4$, respectively.

Here, method 3 is used to find color difference.

For a given pixel $X$ with color $c$,

1. Let the colors of its four precedent neighbors $X_1$ - $X_4$ be $c_1$, $c_2$, $c_3$, respectively.
2. Let the result of sorting the values of $c_1$ - $c_4$ according to the color-ordering to the color-ordering relationship be $c_1'$ - $c_4'$, with $c_1'$ being the largest.

We define a color-mapping function with binary output as follows, which we denote as $f_{cm}$:

$f_{cm}(c, c_1', ..., c_4') = 0$, if $c > c_1'$

$= 1$, if $c_1' > c >= c_2'$

$= 0$, if $c_2' > c >= c_3'$

$= 1$, if $c_3' > c >= c_4'$

$= 0$, otherwise.

(3)

It can be seen that the function output depends on the ordering of the color $c$ of $X$ among those of the four precedent neighbors of $X$.

In addition, to reduce possible quality degradation in the resulting stego-image, the pixels in the cover images are classified into data embeddable and non embeddable ones in this study during the raster-scanning process [2]. Only data-embeddable pixels are used for data hiding; non embeddable ones are skipped.

Let $c$ be the original color of a given pixel $X$ and $c'$ a possible replacement for $c$ in the color palette $P$.

When the color of $X$ is $c$, assume that

1. The corresponding output of the color-mapping function of $X$ is $b$, and
2. The corresponding maximum color difference between $X$ and its four precedent neighbors is $\beta$.

When the color $c$ of $X$ is replaced by $c'$, assume that

1. The corresponding values of $b$ and $\beta$ are changed to $b'$ and $\beta'$, respectively.
2. The number of distinct colors of $X$'s four precedent neighbors is $\alpha$.

A pixel $X$ is defined to be data embeddable if the following three conditions are satisfied:

1. $\alpha$ is larger than a threshold value $T_c$
2. $\beta$ is smaller than a threshold value $T_d$
3. there exists a color $c'$ with the corresponding $b'$ being the inverse of $b$, and the corresponding $\beta'$ being smaller than threshold value $T_d$.

Or equivalently, we define the data embeddability of a pixel $X$ as below.

**X is data embeddable, if $\alpha > T_c$, $\beta < T_d$, and there exists a $c'$ such that $b' \neq b$ and $\beta' < T_d**
X is nonembeddable, otherwise. --- (4)

Here, the first condition, $\alpha > T_c$: This condition is used to ensure that X is located in a reasonably color-abundant region so that pixel color modification due to data embedding will arouse little suspicion. A reason here is that the change of a pixel’s color among a region with a lot of colors presumably is less noticeable. The second condition $\beta < T_d$: This condition is set mainly to avoid pixel color modification at high contrast regions with large $\beta$ values, where sharp lines or edges will appear with higher probabilities. Modification of colors at such regions usually will cause more obvious image content changes, and so reduce the data-hiding effect. The third condition: This condition is set to ensure that X has the ability to embed a bit, and that X is still data embeddable after the data-embedding process.

Because the number of distinct colors of X’s four precedent neighbors is not changed when the color of X is altered, the condition $\alpha > T_c$ will not be changed, and so is not included in the third condition.

3. Implementation

The implementation uses three algorithms: 1. optimal replacement color selection for a given pixel. 2. Secret data-embedding process. 3. Secret data-extraction process

3.1 Data-Embedding Process:

Assume that the secret data S to be embedded in a cover palette image I is a bit stream, denoted as $S = b_1 b_2 \ldots b_n$. The basic data-embedding process is to check each pixel of I in a raster-scanning manner for its data embeddability, and to embed each secret bit $b_i$ of S sequentially into every data-embeddable pixel, until the bit stream of S is exhausted.

During each secret bit-embedding step, if the binary output of the color-mapping function $f_{cm}$ is the same as the secret bit value to be embedded, the color $c$ of the currently checked data-embeddable pixel X is kept unchanged; otherwise, c is replaced with a color $c_{opt}$ called the optimal replacement color for X, selected from the color palette by the following algorithm.

Algorithm 1: Optimal replacement color selection for a given pixel

Input: The color $c$ of the currently checked data-embeddable pixel X, the color palette P, and a secret bit value $b$ to be embedded.

Output: The optimal replacement color $C_{opt}$ for X.

1: Let C denote a set of candidate optimal replacement colors for X, and set C empty initially.

2: Put each color $c_i$ in the color palette P into C if $c_i$ satisfies the following two conditions:
   1. $c_i$ together with the colors of the four precedent neighbors of X as input to the color-mapping function $f_{cm}$ yields a binary output value $b_0$ equal to the secret bit $b$; and
   2. X is still data embeddable when its color is set to $c_i$.

3: Find the color $c'$ among those in C, whose color difference from c is the minimum, i.e., find the $c'$ in C that satisfies the following condition:
   $$|c - c'| = \min |c - c_i|, \ c_i \in C$$

4: Let N denote the subset of C that contains the colors of the four precedent neighbors of X. If the color difference $|c - c'|$ between c and $c'$ is smaller than a preselected threshold $T_v$ or N is empty, then take $c'$ as the desired optimal replacement color $C_{opt}$ for X and stop; otherwise, perform the next step.

5: Find the color $c'$ among those in N, where color difference from c is the minimum i.e., find the $c'$ that satisfies the following condition:
   $$|c - c'| = \min |c - c_i|, \ c_i \in N$$

   take $c'$ as the desired $C_{opt}$ for X; and stop.

The first three steps of the above algorithm aim to select from the color palette a color $c'$ which is “most similar” in color to that of the currently checked pixel X. However, if this color $c'$ differs too much from that of X, use of it as the desired optimal replacement color will cause an obvious visual artifact at X. 4 is performed with the preselected threshold $T_v$ used to avoid such a case. If this case is found to exist. In 5, the “most similar” color found...
from those of the four precedent pixels is taken to be the desired optimal replacement color for the currently checked pixel X.

An important idea behind the Algorithm 1 is that one require the pixel X to be still data embeddable after its color is replaced, as depicted by the second condition in Step 2. This measure facilitates the work of identifying those pixels into which secret bits have been embedded during the secret data-extraction process.

Algorithm 2: Secret data-embedding process
Input: A cover palette image I, and a secret bit stream S = b₁ b₂, …, bₙ to be embedded.
Output: An image I’ with the secret S being embedded.

1: For each secret bit bₖ in S, perform the following steps until all secret bits in S are embedded.
2: Perform a raster scan of image I and check the data embedability of each scanned pixel, until a data-embeddable pixel X is found.
3: Take the color c of X and the sorted colors c₁’-c₄’ of the four precedent neighbors of X as input to the color-mapping function f_cₘ to yield a binary output bit b₀.
4: Check whether the secret bit bₖ is equal to b₀. If so, regard the secret bit bₖ to already exist at X, and go to Step 1 to embed the next bit; otherwise, perform the next step.
5: Find the optimal replacement color c_opt for X by Algorithm 1; substitute the color c of X with c_opt; and go to Step 1 to embed the next bit.

3.2 Data-Extraction Process:

During the data-extraction process, a given stego-image is also processed in a raster-scanning manner.

1. Check the pixel’s data embedability. Only data-embeddable pixels are processed further; all non embeddable ones are skipped.
2. For each data-embeddable pixel, we take its color and those of its four precedent neighbors as input to the color-mapping function f_cₘ and compute the binary output value.
3. If the output is “0”, then the extracted secret bit is taken to be “0”, otherwise, “1”.

The extraction process is simple and so can be performed very fast. The detailed data-extraction process is described in the following as an algorithm.

Algorithm 3: Secret data-extraction process
Input: An input stego-image I’ in which a secret bit stream S = b₁ b₂, …, bₙ was embedded.
Output: The secret bit stream S.
1: Set S as an empty bit stream initially.
2: Perform a raster scan of I’ and execute the following steps until all pixels of I’ are processed.
3: For each pixel X of I’, check whether it is data embeddable. If not, regard no secret bit to exist at X, and go to Step 2; otherwise, perform the next step.
4: Take the color c of X and the sorted colors c₁’-c₄’ of the four precedent neighbors of X as input to the color-mapping function f_cₘ to yield a binary output bit b.
5: Append b to the end of S, and go to Step 2.

4. Testing and Result Analysis

To test the performance of proposed method, the series of experiments were conducted on a collection of 12 palette images. These images were selected to simulate the use of palette images in real-world applications. Accordingly, the pixel is data embeddable only when there are three distinct colors among it’s four precedent neighbors. Maximum color difference between pixel X and the four precedent neighbors is smaller than β. The experimental results shows that secrete data can be embedded without introducing visual artifacts and extracted correctly by the proposed method. Among various images 3 images are shown in Figure 3 – Figure 5. Analysis of the experimentation is shown in Table 1 and Table 2. Table 1 show results of hiding text file in image file and Table 2 shows results of hiding image file in image file.

5. Conclusion and Future Enhancement

The major idea of the proposed data embedding process is to modify colors of data embeddable image pixels [2]. The binary outputs of color mapping function with the
colors of these image pixels as input may be taken as data to be hidden. The color of data embeddable pixel is to be modified to an optimal one, which is selected from color palette and has the least distortion under the color conditions around the pixels. Different from other data hiding methods that consider color difference in secret embedding. This method in addition takes spatial properties of image content into account. Hence, when cover images contain limited colors that are visually uncorrelated, the proposed method can yield embedding results with better visual quality.

It is found that data up to 21% can be hidden in Gif image. It is also observed that percentage of embeddable pixels depends on standard variation of colors in Gif image. It is not related to size of the image and number of colors in Gif image. It is found that as $T_c$ goes on decreasing and $T_d$ goes on increasing, the percentage of embeddable pixels increases. The PSNR depends only on luminance values of the image. PSNR is more if the difference between $T_c$ and $T_d$ is less. The graphical representation gives the performance of software.

6. References:


Authors:

Dr. N. V. Kalyankar: Dr. N. V. Kalyankar is a recognized guide in Computer Science, he is actively involved in the research related activities from 14 years. At present 14 research candidates are working under his guidance in the field of computer Science.

Dr. S. D. Khamitkar: Dr. S.D. Khamitkar is a Associate professor in the School of Computational Science, He has teaching and research experience of 14 years. At present 8 candidates are working under his guidance in the field of Computer Science

P. U. Bhalchandra: S. N. Lokhande:

N. K. Deshmukh:

All are assistant professors and research scholars in computer Science. They have more than 8 years of teaching experience.
### Table 1: The Results of hiding text file in image file.

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Colors</th>
<th>Embeddable pixels (%)</th>
<th>PSNR (dB)</th>
<th>Embeddable pixels (%)</th>
<th>PSNR (dB)</th>
<th>Embeddable pixels (%)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photographic image</td>
<td>200 x 261 = 52200</td>
<td>00.00</td>
<td>60.21</td>
<td>02.18</td>
<td>70.49</td>
<td>00.00</td>
<td>63.63</td>
</tr>
<tr>
<td>Comic image</td>
<td>411 x 286 = 117546</td>
<td>00.00</td>
<td>50.30</td>
<td>01.44</td>
<td>54.69</td>
<td>00.02</td>
<td>83.36</td>
</tr>
<tr>
<td>Text with graphics image</td>
<td>639 x 852 = 544428</td>
<td>00.00</td>
<td>50.30</td>
<td>01.44</td>
<td>54.69</td>
<td>00.02</td>
<td>83.36</td>
</tr>
</tbody>
</table>

### Table 2: The results of hiding image file in image file.

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Colors</th>
<th>Embeddable pixels (%)</th>
<th>PSNR (dB)</th>
<th>Embeddable pixels (%)</th>
<th>PSNR (dB)</th>
<th>Embeddable pixels (%)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photographic image</td>
<td>200 x 261 = 52200</td>
<td>00.00</td>
<td>60.21</td>
<td>02.18</td>
<td>70.49</td>
<td>00.00</td>
<td>63.63</td>
</tr>
<tr>
<td>Comic image</td>
<td>411 x 286 = 117546</td>
<td>00.00</td>
<td>50.30</td>
<td>01.44</td>
<td>54.69</td>
<td>00.02</td>
<td>83.36</td>
</tr>
<tr>
<td>Text with graphics image</td>
<td>639 x 852 = 544428</td>
<td>00.00</td>
<td>50.30</td>
<td>01.44</td>
<td>54.69</td>
<td>00.02</td>
<td>83.36</td>
</tr>
</tbody>
</table>