Secure Steganographic Methods for Palette Images

Jiri Fridrich, Du Rui

Center for Intelligent Systems, Dept. of SS&IE, SUNY Binghamton, Binghamton, NY 13902-6000
{fridrich,bh09006}@binghamton.edu

Abstract. In this paper, we study non-adaptive and adaptive steganographic techniques for images with low number of colors in palette image formats. We have introduced the concept of optimal parity assignment for the color palette and designed an efficient algorithm that finds the optimal parity assignment. The optimal parity is independent of the image histogram and depends only on the image palette. Thus, it can be used for increasing the security of steganographic techniques that embed message bits into the parity of palette colors. We have further developed two adaptive steganographic methods designed to avoid areas of uniform color and embed message bits into texture-rich portions of the carrier image. Both techniques were tested on computer generated images with large areas of uniform color and with fonts on uniform background. No obvious artifacts were introduced by either technique. The last, embedding-while-dithering, technique has been designed for palette images obtained from true color images using color quantization and dithering. In this technique, both the color quantization error and the error due to message embedding are diffused through the image to avoid introducing artifacts inconsistent with the dithering algorithm.

1 Introduction

The purpose of steganography is to hide messages in otherwise innocent looking carriers. The purpose is to achieve security and privacy by *masking the very presence of communication*. Historically, the first steganographic techniques included invisible writing using special inks or chemicals. It was also fairly common to hide messages in text. By recovering the first letters from words or sentences of some innocent looking text, a secret message was communicated. Today, it seems natural to use binary files with certain degree of irrelevancy and redundancy to hide data. Digital images, videos, and audio tracks are ideal for this purpose.

Each steganographic technique consists of an embedding algorithm and a detector function. The embedding algorithm is used to hide secret messages inside a cover (or carrier) document. The embedding process is usually protected by a keyword so that only those who posses the secret keyword can access the hidden message. The detector function is applied to the carrier and returns the hidden secret message. For secure covert communication, it is important that by injecting a secret message into a carrier document no *detectable changes* are introduced. The main goal is to not raise suspicion and avoid introducing statistically detectable modifications into the carrier

document. The embedded information is undetectable if the image with the embedded message is consistent with the model of the source from which the carrier images are drawn. We point out that the ability to detect the presence does not automatically imply the ability to read the hidden message. We further note that undetectability should not be mistaken for invisibility – a concept tied to human perception. At present, the formal theoretical framework for steganography similar to Shannon information theory is still missing. For a comprehensive treatment of this topic, see [1].

The undetectability is directly influenced by the size of the secret message and the format and content of the carrier image. Obviously, the longer the message, the larger the modification of the carrier image and the higher the probability that the modifications can be statistically detected. The choice of the carrier image is also important. Natural photographs with 24 bits per pixel provide the best environment for message hiding. The redundancy of the data helps to conceal the presence of secret messages. Image formats that utilize color palettes provide efficient storage for images with limited number of colors, such as charts, computer art, or color quantized true color images. The palette image format GIF is recognized by all browsers and is widely used over the Internet. Posting a GIF file on one's web page will undoubtedly raise less suspicion than sending an image in the BMP format. Despite their usefulness and advantages, palette images provide a hostile environment for the steganographer. The limited number of palette colors makes the process of secure message hiding a difficult challenge. The most common steganographic technique – the least significant bit embedding (LSB) cannot be directly applied to palette images because too many new colors would be created. Most current steganographic algorithms for palette images introduce easily detectable artifacts in the palette or in the image data [8,9].

On the highest level, the typical palette image format consists of three parts: a header, a palette, and image data or pointers to the palette. The palette contains the RGB triplets of all colors that occur in the image. Secret messages can be hidden either in the palette itself or in the image data. Gifshuffle [10] is a program that uses the *palette order* to hide up to $\log_2(256!)=210$ bytes in the palette by permuting its entries. While this method does not change the appearance of the image, which is certainly an advantage, its security is weak because many image processing software products order the palette according to luminance, frequency of occurrence, or some other scalar factor. A randomly ordered palette is suspicious, which goes against the basic requirement of secure steganography. Also, displaying the image and resaving it may erase the information because the palette may be reordered. An alternative and perhaps more secure approach is to hide encrypted messages in the LSBs of the palette colors. In order to make the message readable from an image with a reordered palette, care needs to be taken during message embedding so that the message is readable at the receiving end. The common disadvantage of all techniques that embed message bits into the palette is a rather limited capacity independent of the image size.

Practical methods should have capacity proportional to the image size, or the number of pixels. Many currently available software tools [3,4,7,10–13] decrease the color depth of the GIF image to 128, 64, or 32 before the embedding starts. This way,

when the LSBs of one, two or three color channels are perturbed, the total number of newly created colors will be at most 256. Thus, it will be possible to embed one, two, or three bits per pixel without introducing visible artifacts into the carrier image. However, as pointed out by Johnson [8,9], the new palettes will have easily detectable groups of close colors. It is thus relatively easy to distinguish images with and without secret messages. It appears that secure schemes should not manipulate the palette but rather embed message bits in the image data.

In the next section, we discuss methods that embed message bits as parities of colors. In Sect. 3, we define the energy of distortions due to message embedding and introduce the concept of optimal parity assignment that minimizes this energy. An efficient algorithm for the optimal parity is presented and the proof of optimality is given. The technique is further extended to multiple pixel embedding. It is shown that the optimal parity assignment is also optimal for multiple-pixel embedding. In Sect. 4, we study adaptive steganographic techniques. Two methods are introduced and their performance is tested on computer generated fractal images. A new technique for palette images obtained through color quantization and dithering of true-color images is described in Sect. 5. In this new dithering-while-embedding technique, the image modifications due to message embedding are diffused through the image in the same way as the quantization error during dithering. Finally in Sect. 6, we summarize the new techniques and conclude the paper by outlining future research directions.

2 Message hiding using the parity of palette colors

One of the most popular message hiding schemes for palette-based images (GIF files) has been proposed by Machado [11]. In her method called EZ Stego, the palette is first sorted by luminance. In the reordered palette, neighboring palette entries are typically near to each other in the color space, as well. EZ Stego embeds the message in a binary form into the LSB of randomly chosen pointers to the palette colors. One can say that this method consists of three steps: parity assignment to palette colors (ordering the palette), random, key-dependent selection of pixels, and embedding message into color parities of the selected pixels. Message recovery is simply achieved by selecting the same pixels and collecting the LSBs of all indices to the ordered palette. This algorithm is based on the premise that close colors in the luminance-ordered palette are close in the color space. However, since luminance is a linear combination of three colors, occasionally colors with similar luminance values may be relatively far from each other.

To alleviate this problem, Fridrich [6] has proposed to hide message bits into the parity bits of closest colors[1]. For the color of each pixel, into which we embed message bits, the closest colors in the palette are searched till a palette entry is found with the desired parity bit. The parity of each color could be assigned randomly or simply by calculating $R+G+B$ mod 2. Because the parity bits are randomly distributed, we will never have to depart from the original color too much. This way,

[1] Using parity for message embedding has previously been proposed by Petitcolas [1] and Crandall [5].

we avoid the problem of occasionally making large changes in color, which will certainly contribute to the undetectability of the message. In the absence of a rigorous security definition for steganography, the following quantities were accepted as measures of security:

1. The distance D between the original and the stego-image

$$D^2 = \sum_{i,j=1}^{M,N} d^2_{ij},$$

   where $d_{ij}^2 = (R_{ij}-R_{ij}')^2 + (G_{ij}-G_{ij}')^2 + (B_{ij}-B_{ij}')^2$ for the (i, j)-th pixel of the original and the stego-image.
2. The maximal color change $\max_{i,j} d_{ij}$.

Both the average power per pixel and the maximal color change for the new technique [6] have decreased 4–5 times when compared to the EZ Stego, which is a significant performance improvement.

In the next section, we investigate the problem of optimal parity assignment for the palette in order to further improve the scheme described in this section.

3 Optimal parity assignment

The parity assignment directly influences the energy of image modifications due to message embedding. Obviously, if close colors are assigned opposite parities, the energy of the modifications will be smaller. A natural question to ask is whether it is possible to further improve the performance by using an optimized palette parity assignment. For a practical method, which does not have access to the original image, the palette parity assignment has to be reconstructable from the modified image at the receiving end.

Let the image palette contain N colors $c_1, c_2, \ldots, c_N$ with parities P_i, $P_i \in \{0,1\}$. The parity assignment determines an isolation s_i for the i-th color (s_i is the distance from color c_i to the closest color with different parity). The colors occur in the original image with frequencies $p_1, \ldots, p_N$, $p_1 + \ldots + p_N = 1$. Provided the message carrying pixels are selected non-adaptively, for a message of length k, approximately kp_i pixels of color c_i will contain message bits. The average square of the distance between the original and the stego-image can be expressed as:

$$\frac{1}{2} kE(P_1,\ldots,P_N) = \frac{1}{2} k \sum_{i=1}^{N} p_i s_i^2 .$$

The quantity E does not depend on the message length and will be called the *energy of the parity assignment*. The optimization problem is to assign parities $P_1, \ldots, P_N$ to

colors c_1, …, c_N so that E is minimal. The following algorithm always finds the optimal parity assignment that minimizes E:

3.1 Algorithm for optimal parity assignment

1. Calculate the distances between all pairs of colors $d_{ij} = |c_i - c_j|$. The distance can be calculated either in the RGB or the YUV space. Set $C = \varnothing$.
2. Order the distances starting from the smallest to the largest, $\{d\} = d_{i(1)\,j(1)} \le d_{i(2)\,j(2)} \le \ldots$.
3. Iteratively repeat step No. 4 until C contains all N colors.
4. Choose the next distance d_{kl} in the ordered sequence $\{d\}$ such that either $c_k \notin C$ or $c_l \notin C$. If more than one d_{kl} is the smallest, randomly choose one. If there is no such d_{kl} this means that C contains all N colors and we are done. If both $c_k \notin C$ or $c_l \notin C$, assign two opposite parities to both k and l. If $c_k \notin C$ and $c_l \in C$, set $P_k = 1 - P_l$. Update $C = C \cup \{c_k\} \cup \{c_l\}$.

It is clear that once a parity of a color is defined, it cannot be changed later by the algorithm. It is also clear that at the end all colors will have assigned parities. What needs to be proved is that the parity assignment has the minimal energy E. We point out that the minimal value of E can occur for more than one parity assignment (see Fig. 1).

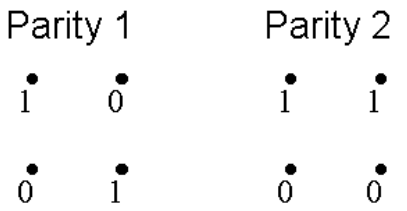

Fig. 1. Example of two different optimal parity assignments

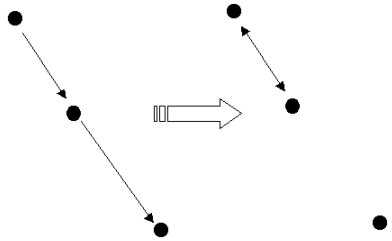

Fig. 2.

Each parity assignment induces a structure of an oriented graph with an arrow pointing from each color (a node) to its closest color with different parity.

Properties of the graph: There is exactly one arrow going out of every node. One node may have more than one incoming arrow. Some nodes may not have any incoming arrows (end nodes). Note that for an optimal parity assignment, when following a sequence of arrows, the distances can never increase because one could simply flip the arrow of the larger distance back to the previous node and decrease the energy E (see Fig. 2).

Proof of the optimality: Suppose that we have a parity assignment P that is optimal. We will show that by following our algorithm, it is possible to reassign the parity to a different parity obtainable with our algorithm while preserving the energy.

Let us assume that while following our algorithm we find a distance d_{ij} with $P_i = P_j$, such that either $c_i \notin C$ or $c_j \notin C$. Let us assume that $c_j \notin C$. We will show that it is possible to change the parity of this color and other colors pointing to this color while the energy E stays the same. Let $c_1^{(1)}, c_2^{(1)}, \ldots, c_{N(1)}^{(1)}$ be the colors (nodes) with arrows pointing to c_j. While flipping the parity of c_j, we also flip the parities of $c_1^{(1)}, c_2^{(1)}, \ldots, c_{N(1)}^{(1)}$. Then we do the same thing with all nodes pointing to one of $c_1^{(1)}, c_2^{(1)}, \ldots, c_{N(1)}^{(1)}$, etc. We have to show that while proceedings in this way we never run into a conflict of having to flip the same node to a 0 and 1 at the same time. This is not possible because such a node would have to have two outgoing arrows. We can end up in an end node, but we also can form a cycle and get back to c_j via a node c_k to which c_j points (see Fig. 2). If $d_{jk} > d_{ij}$, we could cancel the arrow pointing from c_j to c_k and replace it with an arrow from c_j to c_i decreasing the energy by $p_j(d_{jk}-d_{ij})>0$, which is a contradiction with P being optimal. The other inequality, $d_{jk} < d_{ij}$ implies $c_j \in C$, which contradicts the assumption. Therefore, $d_{jk} = d_{ij}$ and the energy E stays unchanged by replacing the arrows. In the case that the "avalanche" of parity flipping would also flip the parity of c_i, we argue as follows. If $d_{il} > d_{ij}$ we can replace the arrow pointing from c_i to c_l with an arrow from c_i to c_j and decrease the energy. If $d_{il} < d_{ij}$ we can follow the arrows on the branch from c_i back to c_j to obtain $c_j \in C$, which is again contradiction with our assumption. Therefore, we must have $d_{kl} = d_{ij}$ and we can replace the arrow from c_i to c_l with an arrow from c_i to c_j while preserving E.

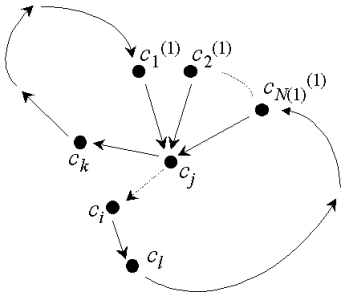

Fig. 3.

As a consequence, we have shown that we can always reassign parities of P in a manner compatible with our algorithm while preserving the energy E. The algorithm and our arguments end when $C = \{c_1, \ldots, c_N\}$. This concludes the proof.

Note that the optimal parity assignment *does not* depend on the frequency of occurrence of pixels, $p_1, \ldots, p_N$. This somewhat contradictory and surprising result enables us to calculate the optimal parity assignment from the modified image without accessing the original image. We just need to order the palette before applying the algorithm (one can use the alphabetical RGB order, for example). If any random decisions are made during the algorithm run, we need to seed a PRNG with the same seed, as well. Another possibility would be to agree on a fixed order in the sequence $\{d\}$, the parity assignment for the first pair, and a rule for assigning the parity when both new nodes $c_i \notin C$ and $c_j \notin C$. Either way, the optimal parity can be utilized for decreasing the energy of modifications in images.

Numerical experiments indicate that when using the optimal parity as opposed to the parity $R+G+B$ mod 2, the energy E is decreased by 25–35% depending on the image.

3.2 Multiple-pixel embedding

It is possible to further decrease the energy of modifications due to message embedding by using clusters of q pixels for one message bit rather than single pixels. The image is first divided using a PRNG into disjoint clusters of random q pixels and the message bit is encoded as a parity of the sum of all q pixel parities. This has the benefit that in order to change the parity of the whole sum one can select a pixel with the smallest isolation among all q pixels. As a consequence, the energy of modifications due to message embedding must decrease. The previous method is a special case of this multiple pixel encoding with $q = 1$.

Below, we show that the optimal parity for this method is the same as for the single pixel embedding. The energy E of the parity assignment is defined in a similar manner

$$E(q) = \frac{1}{2}\sum\nolimits_{i=1}^{N} p_i^{(q)} s_i \ ,$$

where $p_i^{(q)}$ is the probability that among randomly chosen q pixels in the image the one with the smallest isolation is the i-th color. If $p_i^{(q)}$ does not depend on s_i, the optimal parity is again only a function of the palette and not of the image. To calculate the probabilities $p_i^{(q)}$, we rearrange the colors c_i so that their isolations form a non-decreasing sequence. It can easily be shown that

$$p_i^{(q)} = \left(\sum\nolimits_{j\geq i}^{N} p_j^{(q)}\right)^q - \left(\sum\nolimits_{j>i}^{N} p_j^{(q)}\right)^q .$$

Because the probabilities $p_i^{(q)}$ do not depend on the isolations, s_i, the optimal parity for single pixel embedding is also optimal for multiple pixel embedding. The energy $E(q)$ as a function of q is depicted for two test images "Fox" and "Mandrill" in Figs. 4–7. Observe that even with $q = 2$, the energy of modifications could decrease by more than a third. For the test image "Fox", the energy went down to 50% of its original value for $q = 3$. This observation suggests that the multiple pixel embedding technique is a useful and significant security improvement.

Fig. 4. Test image "Fox"

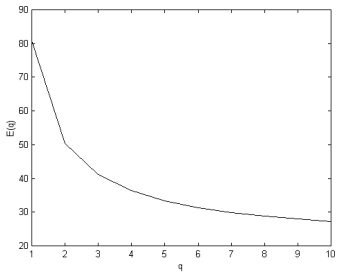

Fig. 5. Energy of modifications as a function of the number of pixels q for the test image "Fox"

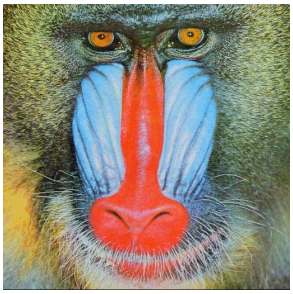

Fig. 6. Test image "Mandrill "

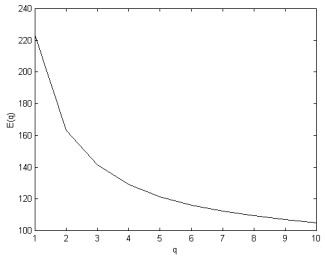

Fig. 7. Energy of modifications as a function of the number of pixels q for the test image "Mandrill"

In this section, we have shown that any method in which pixels are selected in a random (non-adaptive) manner and in which the message bits are embedded by changing the parity of pixels (the analog of LSB encoding for palette images), cannot perform better than our method. This is because our method uses the best parity assignment possible. The result is general and holds for both single pixel embedding and multiple pixel embedding. The next section is devoted to adaptive methods for message embedding to avoid areas of uniform color and avoid creating detectable artifacts in singular images with large areas of uniform color.

4 Adaptive methods

In this section, we explore the idea of an adaptive steganographic technique that would introduce less detectable artifacts into the carrier image by adapting the message embedding technique to the content of the carrier image. *Non-adaptive* steganographic techniques are techniques in which the modifications due to message embedding are uncorrelated with image features. Examples are LSB encoding in randomly selected pixels, message embedding by randomly modulating pixel values or frequency bins in a fixed band, etc. In *adaptive steganography* the modifications

are correlated with the image content (features). For steganography, this could mean that the pixels carrying message bits are selected adaptively and depend on the image. For example, we could avoid areas of uniform color and select only pixels with large local standard deviation. This however creates a problem with message recovery if the original unmodified image is not available. We have to be able to extract the same set of message carrying pixels at the receiving end from the modified image.

The capacity of adaptive schemes is necessarily image dependent and, in some cases, it is not even possible to calculate the capacity before the actual embedding starts. But this is a price we have to pay for increased security. There are several different ways how a non-adaptive scheme can be turned into an adaptive one. For example, the message carrying part and the part that determines the pixel selection do not interact. For example, one can calculate the local standard deviation (STD) from the 7 most significant bits and embed message bits into the LSB. This approach is plausible only for high color images and cannot be adopted for palette images.

Method 1. The image is divided into disjoint blocks (for example 3×3 blocks) completely covering the image. At most one bit will be assigned to each block (either the LSB of the middle pixel or its parity, or the parity of the whole block, etc.). A threshold for local STD is selected. A pseudo-random non-intersecting walk over the blocks is generated from a secret key. If the local STD of a block is above the threshold AND stays above the threshold after message embedding, we select the block for message embedding. If the STD falls below the threshold after message embedding, we make the change anyway but do not include the block for message embedding, and continue message embedding with the same bit in the next block. This process will guarantee that at the receiving end it is enough to regenerate the same random walk over the blocks and read the message bits only from blocks whose STD is above the threshold. Note that the local standard deviation can be replaced with a different quantity, such as the number of colors in the block. Actually, our experiments show that the number of colors works better than the standard deviation for computer generated low-color images.

The advantage of Method 1 is that any pixel(s) in the block can be modified, which will generally lead to smaller modifications especially for images with low color depth. The disadvantage is that it is not possible to say whether or not a message of a given length will fit into the image before one actually starts the embedding process. Another disadvantage is somewhat decreased capacity. At most one bit is inserted into each block, and only blocks with local STD above the threshold are considered for embedding. It is, however, only natural that more secure schemes will have smaller capacity, while "greedy" schemes emphasizing the capacity will offer less security. It will be up to the end user to decide about the priorities.

We tested Method 1 for several fractal GIF images[2] shown in Figs. 8–10. The image in Fig. 8 depicts a Julia set generated on a computer. Julia sets are fractal sets

[2] Images provided courtesy of J. C. Sprott, Fractal Gallery, http://sprott.physics.wisc.edu/fractals.htm.

parametrized using complex numbers. The image is a bad carrier image for steganography for at least three reasons:

1. There are large areas with flat color. Such areas should be avoided otherwise detectable artifacts will be created.
2. The image has been computer generated and has a known complicated inner structure common to all Julia sets. For example, the bands of close colors gradually changing from white to dark red form lines of constant potential and have to form simply connected sets. Therefore the band boundaries, too, should be avoided while embedding a message.
3. The title of the image contains fonts and any subtle change to those fonts will be easily detected.

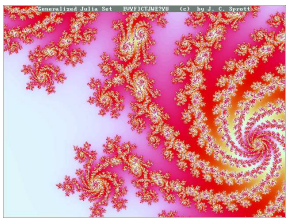

Fig. 8. Fractal carrier image 640×480, 195 colors

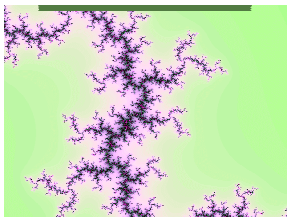

Fig. 9. Fractal carrier image, 636×476, 236 colors

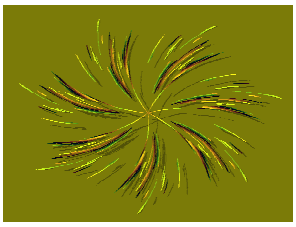

Fig. 10. Fractal carrier image 636×476, 183 colors

A non-adaptive technique, no matter how good, will always create some easily detectable artifacts (see Figs. 9–10). On the one hand, we are tempted to say that the image should not be used for secure steganography. On the other hand, we intuitively feel that those areas with complex structure *can* hold some additional information without causing suspicious artifacts. A smart adaptive technique should be able to recognize which portions of the image can be used for data hiding.

A freeware software utility called MandelSteg [7] uses fractal images for generating carrier images. The parameters of the Julia set form a portion of the secret key. The algorithm is naïve, non-adaptive, and provides very poor security. First, the fact that the software that generates the images is available to anybody enables an efficient search for the parameters of the Julia set by anybody skilled in the art of fractals. Second, the fractal images have complicated inner structure that follows strict rules. Any non-adaptive technique must introduce severe and easily detectable artifacts, such as those depicted in Figs. 11–12.

A version of Method 1 has been used with the local statistics corresponding to the number of different colors in a pixel's neighborhood. We embed a bit of information whenever there are at least three different colors in the 3×3 neighborhood. This in fact will guarantee that there will be no artifacts in the potential lines around the Julia set and no artifacts in the fonts (white characters on uniform background). A detailed inspection of the carrier image with over 1000 embedded bits did not reveal any suspicious artifacts.

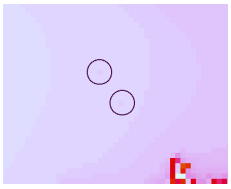

Fig. 11. Artifacts in equipotential bands

Fig. 12. Artifacts in the fonts

Fig. 13. Fonts on a complex background

Method 2. Method 1 performs satisfactorily even for difficult carrier images, such as the images in Figs. 8–10. However, the price we paid was a capacity decreased by a factor of more than 9. Method 2 has been designed with the intent to put on average more bits per pixel while retaining the adaptive property. The message embedding process starts with pseudo-randomly selecting the message carrying pixels. Before we explain the selection process, we introduce some definitions. A 2×2 block is good if it has at least three different colors. In all other cases, the block is bad. A pixel is termed good if all four 2×2 squares that contain that pixel are good. The message carrying pixels are selected pseudo-randomly from the set of all good pixels in the image. Then, all palette colors are assigned the optimal parity using the algorithm from Sect. 3. Finally, we walk through the selected pixels and compare the parity of each pixel to the message bit. If there is a match, we do not modify the pixel and move to the next pixel. If there is no match, we modify the pixel color by searching through the palette for closest neighbors with the correct parity so that after the change all four blocks containing the pixel P stay good. Then, we again move to the next pixel. The embedding procedure guarantees that the set of good blocks for the original image and for the modified image are identical. This enables the detection algorithm to recover the message from color parities by going through the good pixels in the same pseudo-random manner as during the embedding.

The capacity of Method 1 and 2 is compared in Table 1. We ran a series of experiments for different seeds for the PRNG that was used to generate the pseudo-random walk and calculated the standard deviation of the capacity. All three test images shown in Figs. 8–10 were used for this experiment. As expected, Method 2 has significantly higher capacity than Method 1.

To find out the energy of modifications due to message embedding, we used the same three test images and for different seeds we calculated the average MSE distortion per pixel. The results are shown in Table 2. We have also experimented with different combinations of the block size (2×2 and 3×3) and different number of colors (2, 3 or 4) for the goodness criterion. The combination of 2×2 pixels and at least three colors gave us the best results with no perceptible artifacts.

Table 1. Comparison of capacity for Method 1 and 2

	Capacity as a percentage of the total number of pixels

		Figure 8	Figure 9	Figure 10
Method 1		6.51~6.53%	3.16~3.18%	1.11~1.12%
Method 2	2 colors	55.64%	25.92%	10.37%
	3 colors	43.63%	16.94%	1.32%

The mean-square-error between the original and the stego-image for all three fractal test images with embedded different size of data is illustrated in Tables 2–4.

Table 2. MSE for Fig. 8 and different message length

Size of embedded data	2371 Bytes	1331 Bytes	645 Bytes	132 Bytes
Method 1	0.26 0.005	0.15 0.003	0.07 0.002	0.017 0.001
Method 2	0.54 0.01	0.30 0.007	0.15 0.015	0.039 0.011

Table 3. MSE for Fig. 9 and different message length

Size of embedded data	1019 Bytes	739 Bytes	466 Bytes	132 Bytes
Method 1	0.15 0.005	0.11 0.005	0.07 0.003	0.02 0.002
Method 2	0.35 0.009	0.25 0.01	0.16 0.006	0.05 0.004

Table 4. MSE for Fig. 8 and different message length

Size of embedded data	376 Bytes	285Bytes	192 Bytes	132 Bytes
Method 1	0.05 0.004	0.04 0.003	0.03 0.002	0.02 0.002
Method 2	0.40 0.042	0.29 0.02	0.20 0.024	0.15 0.02

Tables 2–4 show that the for images in Figs. 8 and 9 the average power per pixel due to message embedding is about 4–5 times bigger for Method 2 than for Method 1. This can be attributed to the fact that in Method 1 we can select the message carrying pixel as the one with the smallest isolation out of nine pixels. Also, in Method 2 occasionally large modifications may result due to the fact that all eight 2×2 squares must stay good after embedding. For image in Fig. 10, the difference is even more pronounced. Both methods can embed messages into computer generated fractal images without introducing any artifacts into regions of uniform color, areas with equipotential lines around the Julia sets, and aliased fonts on uniform background. Method 1 provides better security because it introduced smaller power per pixel due to message embedding. Method 2 has higher capacity but may occasionally introduce large changes in color.

5 Embedding while dithering

As the last steganographic method of this paper, we describe a technique appropriate for message hiding in palette images obtained from true-color images using color quantization and dithering. Given a true color image, we first derive the color palette using some of the standard color quantization algorithms [2]. If the true color image is

not known and only its quantized form is available, we could increase the color depth using for example the method described in [14]. We continue by calculating the optimal parity assignment for the palette. Then we pseudo-randomly select the message carrying pixels in the image (in a non-adaptive manner). Finally, we perform the quantization and dithering by scanning the image by rows. For a non-message carrying pixel, we perform the standard quantization and dithering step. When we come to a message carrying pixel, we quantize its color to the closest palette color *with the right parity*. This way, both the quantization error and the error due to message embedding will be scattered and diffused through the whole image. We conjecture that the artifacts introduced by this method will be less detectable because they should be consistent with the dithering mechanism.

We have tested the method on 24 bit scans of photographs. As expected, such images typically provide enough texture and color variations, and even when as much as 50% of all pixels in the image have been used for embedding, we could not identify any visible artifacts or suspicious patterns. The method performed satisfactorily even for "singular" images, such as the cartoon of the Tweety Bird shown in Fig. 14. The image is a true color cartoon on a background that gradually changes color. We have again embedded a message of length equal to one half of the number of all pixels. Fig. 15 is a close-up of the original Tweety's eye and beak, and Figs. 16 and 17 show the same close-up for the non-adaptive naïve method with non-optimal palette, respectively. The non-optimized, non-adaptive method performs poorly and one can identify a suspicious pattern in Tweety's eye. On the other hand, the edges and contrast of the original picture has been more or less preserved. The dithering-while-embedding method does not introduce suspicious patterns but the image edges (see the black lines are somewhat blurred due to the diffused error. The color variations in Tweety's beak have also been flattened out.

6 Conclusion and future directions

In this paper, we study non-adaptive and adaptive steganographic techniques for images in palette image formats. We have introduced the concept of optimal parity assignment and designed an efficient algorithm that finds the optimal parity assignment. The optimal parity is independent of the image histogram and depends only on the image palette. Thus, it can be used for increasing the security of steganographic techniques that embed message bits into the parity of palette colors. We have shown that the optimal palette improves the average power per pixel due to message embedding by 25–35% (for method introduced in [6]). The optimal parity is also optimal for multiple pixel embedding.

We have developed two new methods for adaptive message embedding in palette images. The techniques tend to avoid areas of uniform color and embed message bits into texture-rich portions of the carrier image. Both techniques utilize the optimal parity assignment. The first technique embeds one message bit into a group of 3×3 pixels. The second technique has higher capacity, but provides less security when measured by the average distortion power per pixel. Both techniques were tested on

computer generated images with large areas of uniform color and with fonts on uniform background. No obvious artifacts were introduced by either technique. An image with simple vertical bands of thickness of at least 3 with different colors would be classified by both techniques as a zero capacity image. This intuitively corresponds to our feeling that such an image should not be used as a carrier image and the algorithms work as we would expect. At this point, we stress that it is almost impossible to design an algorithm that would work well on all low-color images, especially computer generated images or images with well-defined inner structure.

Fig. 14. Test image Tweety Bird

Fig. 15. Original

Fig. 16. Non-adaptive non-optimized

Fig. 17. Dithering-while-embedding

For example, an image with fonts of uniform color on a complex background, rather than a uniform background as in Fig. 3, will not be handled by our adaptive algorithm correctly (see Fig. 13). The border pixels of fonts may be changed. In a situation like this, it becomes increasingly difficult to automatize the process of secure adaptive selection of pixels. Image understanding and interpretation of image features comes into play. While a human can easily recognize that a pixel is actually a dot above the letter "*i*" and thus must not be changed, it would be very hard to design an algorithm that would recognize this automatically.

The last technique described in this paper has been designed for embedding large messages into palette images obtained from true color images using color quantization and dithering. The basic idea behind this embedding-while-dithering method is to dither both the color quantization error and the error due to message embedding to avoid introducing artifacts inconsistent with the dithering algorithm. We argue that the stego-image will correspond to an image obtained by quantizing and dithering a slightly noisier version of the original image, thus making the stego-image free of artifacts incompatible with dithering.

Our future research effort will be directed towards a more formal approach to adaptive message embedding including estimates for capacity. The dithering-while-

embedding method could also be improved by making it adaptive to the image content.

Acknowledgements

The work on this paper was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under a grant number F30602-98-C-0009. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U. S. Government.

References

1. Andersen, R.J., Petitcolas, F.A.P., On the limits of steganography. *IEEE Journal of Selected Areas in Communications, Special Issue on Copyright and Privacy Protection* **16** No.4 (1998) 474–481.
2. Balasubramanian, R., Bouman, C.A., Allebach, J.P.: Sequential Scalar Quantization of Color Images. Journal of Electronic Imaging **3** No.1 (1994) 45–59.
3. Black W. (anonymous): StegoDos. ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/ stegodos.zip
4. Brown, A.: S-Tools. http://idea.sec.dsi.uimi.it/pub/security/crypt/code/s-tools3.zip
5. Crandall, R.: Some notes on Steganography. Posted on Steganography Mailing List, December (1998).
6. Fridrich, J.: A New Steganographic Method for Palette-Based Images. Proc. of the IS&T PICS conference, April 1998, Savannah, Georgia (1998) 285–289.
7. Hastur, H.: Mandelsteg. http://idea.sec.dsi.uimi.it/pub/security/crypt/codev (1994)
8. Johnson, N.F., Jajodia, S.: Steganography: Seeing the Unseen." IEEE Computer, February 1998 (1998) 26–34.
9. Johnson, N.F., Jajodia, S.: Steganalysis of Images Created Using Current Steganography Software. Proc. 2nd Workshop on Info Hiding, April 1998, Portland, Oregon (1998).
10. Kwan, M.: Gifshuffle. http://www.darkside.com.au/gifshuffle/
11. Machado, R.: EZ Stego. http://www.stego.com/
12. Maroney, C.: Hide and Seek. ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/hdsk1.0b.zip
13. Nelson, L.: Gif-It-Up. http://www.cs.cf.ac.uk/User/L.Nelson
14. Schmitz, B.E., Stevenson, R.L., Color Palette Restoration. Graphical Models and Image Processing **57** No.5 (1995) 409–419.