Expert Systems with Applications 39 (2012) 11517–11524

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A novel technique for image steganography based on a high payload method and edge detection

Anastasia Ioannidou [a], Spyros T. Halkidis [b,*], George Stephanides [b]

[a] Department of Applied Informatics, University of Macedonia, Egnatia 156, GR-54006, Greece
[b] Computational Systems and Software Engineering Laboratory, Department of Applied Informatics, University of Macedonia, Egnatia 156, GR-54006, Greece

ARTICLE INFO

Keywords:
Image steganography
High payload techniques
Hybrid edge detection
Information hiding
Image processing

ABSTRACT

Image steganography has received a lot of attention during the last decade due to the lowering of the cost of storage media, which has allowed for wide use of a large number of images. We present a novel technique for image steganography which belongs to techniques taking advantage of sharp areas in images in order to hide a large amount of data. Specifically, the technique is based on the edges present in an image. A hybrid edge detector is used for this purpose. Moreover, a high payload technique for color images is exploited. These two techniques are combined in order to produce a new steganographic algorithm. Experimental results show that the new method achieves a higher peak signal to noise ratio for the same number of bits per pixel of embedded image.

1. Introduction

Steganography, which means covered writing, is an art that traces back to ancient times (Petitcolas, Anderson, & Kuhn, 1999). It separates itself from cryptography from the fact that one of its basic aims is that the fact that a secret message is hidden, in contrast to cryptography, should be not known to anybody but the communicating entities. Image steganography (Morkel, Eloff, & Olivier, 2005; Queirolo, 2011), where information is embedded within an image has been widely used during the last decade due to the lowering of the cost of image storage and communication and also due to the weaknesses of the human visual system (HVS).

Regarding the terminology related to image steganography, we note that the original image without the embedded secret message is termed as cover or host image, while the image resulting from embedding this message is termed as stego image. The secret message can be plain text, images, audio or video. After the embedding process, the stego image should look identical to the cover image and be resistant to standard analysis in order to avoid raising suspicion. A stego key is usually used in the embedding process to enhance security since only the person who knows it will be able to extract the secret message. The term payload is used to describe the size of the secret message that can be embedded in a particular image.

Image steganography methods can be separated into two categories: spatial domain and frequency domain based methods. In the first case, the secret message is embedded directly in the intensity of the pixels, while in the second case, images are first transformed to frequency domain and then, the secret message is embedded in the transform coefficients.

Many image file formats, such as jpeg, bmp, and gif, have been used so far in the literature for image steganography. 8-bit and 24-bit images are the most typical, the first due to their small size and the second due to the high payload they offer and to the fact that the large number of colors they contain make the changes from the secret message undetectable from the human visual system.

In this paper we present a novel technique for image steganography which is based on a high payload method for color images (EL-Emam, 2007) and another high payload steganography mechanism using a hybrid edge detector (Chen, Chang, & Hoang Ngan Le, 2010).

The rest of the paper is organized as follows: Section 2 presents related work, Section 3 describes the proposed method, Section 4 presents the experimental results in terms of peak signal to noise ratio and bits per pixel (bpp), Section 5 presents an experiment, where the number of embedded bits in edge pixels is gradually increased and possible visual attacks are investigated. Finally, Section 6 provides some conclusions and proposes future work.

2. Related work

The most obvious method to hide information within an image is the least significant bit (LSB) technique. In this method, information is embedded in the least significant bit of every pixel

* Corresponding author.
E-mail addresses: ioananas@csd.auth.gr (A. Ioannidou), halkidis@java.uom.gr (S.T. Halkidis), steph@uom.gr (G. Stephanides).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.02.106

in the image. However it has been shown (Fridrich, Goljan, & Du, 2001) that this technique does not resist statistical attacks. More recent techniques (Chang & Tseng, 2004; Chen et al., 2010; Thien & Lin, 2003; Wang, Wu, Tsai, & Hwang, 2008; Wu & Tsai, 2003) exploit the fact that more information can be hidden in sharp areas than in smooth areas, without the embedded image to be detectable.

In Wu and Tsai (2003) a steganographic method for images by pixel value differencing is presented. The cover image is partitioned in non-overlapping blocks of two consecutive pixels. A difference value is calculated from the values of the two pixels in each block. All possible difference values are classified into a number of ranges. The selection of the range intervals is based on the characteristics of the human vision's sensitivity to gray value variations from smoothness to contrast. The difference value is then replaced by a new value to embed the value of a substream of the secret message. The number of bits which can be embedded in a pixel pair is decided by the width of the range that the difference value belongs to. They use the differences of the gray values in the two-pixel blocks of the cover image as features to cluster the blocks into a number of categories of smoothness and contrast properties. Different amounts of data can be embedded in different categories according to the degree of smoothness or contrast. The main observation is that changes in the gray values of pixels in smooth areas in images are more easily noticed by the human eyes. A large difference value indicates that we examine an edged area. The image is scanned in a zigzag manner. If the difference value d is close to 0, this indicates that we are located in an extremely smoothed block. Conversely a value of d close to -255 or 255 shows that we are located in a sharply edged block. Only the absolute values of d can be considered (0 through 255) and classified into a number of contiguous ranges say R_i, where i = 1, 2, ..., n. The lower and upper values of R_i are denoted by l_i and u_i, respectively, where l_1 is 0 and u_n is 255. The widths of the ranges which represent the difference values of smooth blocks are chosen to be smaller, while those which represent the difference values of edged blocks are chosen to be larger. That is, ranges with smaller widths are created when d is close to 0 and ones with larger widths are created when d is far away from 0 for the purpose of yielding better imperceptible results. A difference value which falls in a range with index k is said to have index k. In the proposed method some bits of the secret message are embedded into a two-pixel block by replacing the difference value of the block with one with an identical index. The number of bits that can be embedded is given by:

$$n = \log_2(u_k - l_k + 1). \tag{1}$$

The new difference d' is given by:

$$d' = \begin{cases} l_k + b, & \text{if } d \geqslant 0, \\ -(l_k + b), & \text{if } d < 0, \end{cases} \tag{2}$$

where b is the value of the sub-streams. The embedding process is finished when all the bits of the secret message are embedded.

The inverse calculation for computing (g'_i, g'_{i+1}) from the original gray values (g_i, g_{i+1}) of the pixel pair is given by:

$$f((g_i, g_{i+1}), m) = (g'_i, g'_{i+1}) = \begin{cases} g_i - ceiling_m, g_{i+1} + floor_m, \\ \text{if } d = \text{is an odd number}, \\ g_i - floor_m, g_{i+1} + ceiling_m, \\ \text{if } d = \text{is an even number}, \end{cases} \tag{3}$$

where $m = d' - d$, $ceiling_m = \lceil \frac{m}{2} \rceil$ and $floor_m = \lfloor \frac{m}{2} \rfloor$. Care is also taken of boundary values. For the inverse process of extracting the original gray values from the stego-image, we note that the values which are computed from $f((g^*_i, g^*_{i+1}), u_k - d^*)$ (where d^* is the difference of the two gray values which have index k) are identical to the gray values which were computed in the embedding process. A related proof is given in the paper. The gray values of the stego-image are used to determine the index k. The value b, which was embedded in the two-pixel block is extracted by using the equation:

$$b = \begin{cases} d^* - l_k, & \text{if } d^* \geqslant 0, \\ -d^* - l_k, & \text{if } d^* < 0. \end{cases} \tag{4}$$

Note that in the recovery of the secret message from the stego-image using the previously described extraction process, there is no need of referencing the cover image.

A technique not directly but indirectly related to our context, by a method described later, (Wang et al., 2008) is analyzed by Thien and Lin (2003), who illustrate a steganographic method based on the modulus function. Suppose that people wish to embed the i-th digit x_i ($0 \leqslant x_i < m$; $m = 2^k$) of the data into a pixel with a gray value of y_i ($0 \leqslant y_i \leqslant 255$) in the cover image. Assume that

$$y_i = m \times t_i + b_i, \tag{5}$$

where $b_i = y_i \bmod m$, $t_i = \lfloor y_i / m \rfloor$.

The resulting gray value y'_{iLSB} replacing y_i is

$$y'_{iLSB} = (y_i - b_i) + x_i. \tag{6}$$

In later days the embedded data digit x_i can be extracted from y'_{iLSB} by

$$x_i = y'_{iLSB} \bmod m. \tag{7}$$

The resulting gray value y'_i generated by the proposed method of Thien and Lin is given by

$$y'_i = y_i - b_i + x_i + lm, \tag{8}$$

where l is an integer suitably chosen from 0, 1, -1. m kinds of symbols are used in the data. The ith digit x_i ($0 \leqslant x_i < m$) of the data is embedded into the pixel with a gray value of y_i ($0 \leqslant y_i \leqslant 255$) in the cover image. First $b_i = y_i \bmod m$, $d_i = x_i - b_i$.

$$d'_i = \begin{cases} d_i, & \text{if } \left(-\lfloor \frac{m-1}{2} \rfloor\right) \leqslant d_i \leqslant \lceil \frac{m-1}{2} \rceil, \\ d_i + m, & \text{if } (-m+1) \leqslant d_i < \left(-\lfloor \frac{m-1}{2} \rfloor\right), \\ d_i - m, & \text{if } \lceil \frac{m-1}{2} \rceil < d_i < m. \end{cases} \tag{9}$$

Finally $y'_i = y_i + d'_i$ is used as the resulting gray value to replace y_i

$$y'_i = \begin{cases} y_i + d'_i + m, & \text{if } y_i + d'_i < 0, \\ y_i + d'_i - m, & \text{if } y_i + d'_i > 255. \end{cases} \tag{10}$$

The embedded information can be extracted as $x_i = y'_i \bmod m$ just as with the LSB method.

A technique that combines pixel value differencing with the modulus function is proposed by Wang et al. (2008). The remainder of the two consecutive pixels can be computed using the modulus operation and then secret data can be embedded into the two pixels by modifying their remainder. There is an optimal approach to alter the remainder so as to greatly reduce the distortion caused by the hiding of the secret data. Instead of the difference value the proposed scheme modifies the remainder of two consecutive pixels $P_{(i,x)}$ and $P_{(i,y)}$ for better stego-image quality. The method is divided into various steps.

Step 1: Given a sub-block F_i composed of two continuous pixels $P_{(i,x)}$ and $P_{(i,y)}$ from the cover image, obtain the difference value d_i, the sub-range R_j, such that $R_j \in [l_j, u_j]$, the width $w_j = u_j - l_j + 1$, the hiding capacity t_i bits and the decimal value t'_i of t_i for each F_i by using Wu and Tsai's scheme.

Step 2: Compute the remainder values $P_{rem(i,x)}$, $P_{rem(i,y)}$ and $F_{rem(i)}$ of $P_{(i,x)}$ and $P_{(i,y)}$ and sub-block F_i respectively by using the following equations:

$$P_{rem(i,x)} = P_{(i,x)} \bmod t'_i, \quad (11)$$

$$P_{rem(i,y)} = P_{(i,y)} \bmod t'_i, \quad (12)$$

$$F_{rem(i)} = (P_{(i,x)} + P_{(i,y)}) \bmod t'_i. \quad (13)$$

Step 3: Embed t_i bits of secret data into F_i by altering $P_{(i,x)}$ and $P_{(i,y)}$ such that $F_{rem(i)} = t_i$.

Care of the boundary values must also be taken.

In the recovery process the secret data can be extracted without using the cover image. However the original range table R designed in the embedding phase must be used in order to figure out the embedding capacity for each sub-block F_i. Given a sub-block F_i with two consecutive pixels from the stego-image with their pixel values being $P_{(i,x)}$ and $P_{(i,y)}$, respectively, the difference value d_i can be derived as their absolute difference. Each F_i can be related to its optimal sub-range R_j from the original table R according to the difference value d_i. Hence, we can compute the width of the sub-range by $w_j = u_j - l_j$, and the number of bits t_i can be extracted from F_i. Eventually the remainder value of F_i can be computed and the remainder value $F_{rem(i)}$ can be transformed into a binary string with the length t_i. For example, assume that two consecutive pixel values of the stego-image are $P_{(i,x)} = 34$ and $P_{(i,y)} = 33$, and the hidden capacity is 3 bits (i.e. $t_i = 3$). $F_{rem(i)}$ can be gained by $(33 + 34) \bmod 2^3 = 3$. Convert the remainder value 3 into a binary string whose length is 3 and then we have $3_{(10)} = 011_{(2)}$.

In Chang and Tseng (2004) a steganographic method which exploits side information and edge image areas is presented. The concept of side match, which is proposed by Kim (1992), uses the correlation between the upper and left neighboring blocks in order to achieve higher compression. In the proposed method, side information of upper P_U and left P_L neighboring pixels (two sided-match steganography) is used to calculate a difference value d. The message is embedded in the cover image in raster-scan order except for the first row and the first column. The difference value d is calculated as:

$$d = (g_u + g_l)/2 - g_x, \quad (14)$$

where g_x is the gray value of an image pixel P_x and g_u, g_l are the gray values of its upper and left pixels, respectively. The number of bits which can be embedded in each pixel depends on d. If d has values -1, 0 or 1, which means that the pixel is in a smooth area of the image, then only one bit of the secret message is embedded in the LSB of the particular pixel. In any other case, the number of bits which can be embedded in the pixel is computed as:

$$n = log_2|d|, \quad \text{if } |d| > 1. \quad (15)$$

A sub-stream of n bits is extracted from the secret message and is converted to integer b. A new difference value is, then, estimated as follows:

$$d' = \begin{cases} 2^n + b, & \text{if } d > 1, \\ -(2^n + b), & \text{if } d < 1. \end{cases} \quad (16)$$

The new value of the pixel P_x is, eventually, calculated as:

$$g_x = (g_u + g_l)/2 - d'. \quad (17)$$

If the new value g_x goes out of boundary of the range [0 255], this pixel is not used for embedding.

The extraction of the embedded message is simple. The stego-image's pixels are read in a raster-scan order except for the first row and column and a difference value (d^*) is computed for each one of them. If d^* has values -1, 0 or 1, then one bit of secret data is extracted from the LSB of the pixel. Otherwise, n bits are extracted as in the embedding process. The embedding value b is extracted according to the following equation:

$$b = \begin{cases} d^* - 2^n, & \text{if } d^* > 1, \\ -d^* - 2^n, & \text{if } d^* < 1. \end{cases} \quad (18)$$

Value b is then converted to its corresponding binary string with n bits length. The original image is not required in the extraction process.

In order to make a better estimation of the new value g_x of the image pixels after the embedding, three-sided and four-sided side match methods can be used. These methods take advantage not only of the upper and left pixels but of the bottom and right as well. The embedding and extraction equations in these cases are identical to those of two-sided side match steganography.

Regarding the steganographic algorithm for embedding the secret message we illustrate it using a specific example. The image is first separated into blocks of n pixels. The suggested values for n are 3, 4 and 5. The n-pixel block is separated into two categories corresponding to non-edge pixels category and edge pixels category. Each cover pixel in the first category contains 'x' secret message bits using the LSB substitution technique. Each cover pixel in the second category contains 'y' secret message bits using the LSB substitution technique. Usually x is chosen as 1 or 2. y is chosen as 3, 4 or 5. Consider an image A having four pixels as [10101010], [10000000], [11111100], [00001111] corresponding to P_1, P_2, P_3 and P_4 with the secret message S = '0110101'. Consider that P_2 and P_4 are edge pixels while P_4 is a non-edge pixel. Then the status of P_2, P_3 and P_4 is 101. Replace 3 LSBs in pixel P_1 with '101'. So P_1 becomes [10101101]. Suppose that x and y are 1 and 3, respectively. Replace three LSBs in pixel P_2 with 3 secret message bits. Replace one LSB in pixel P_3 with one secret message bit. Replace three LSBs in pixel P_4 with three secret message bits. Then P_2 becomes [10000011], P_3 becomes [11111100] and P_4 becomes [00001101]. Considering the inverse procedure, suppose $n = 4$, $x = 1$ and $y = 3$, as before. Also, suppose we have a stego-image A' having four pixels as [10101101], [10000011], [11111100], [00001101] corresponding to four pixels P'_1, P'_2, P'_3 and P'_4. Obtain $(n - 1) = 3$ LSBs in the first pixel, we get three bits as '101'. Thus the second and fourth pixels are edge pixels. We will extract three LSBs from the pixel P'_2 and the pixel P'_4. We will also extract one LSB from the pixel P'_3. The extracted bits from the pixel P'_2 are '011'. The extracted bit from the pixel P'_3 is '0'. The extracted bits from the pixel P'_4 are '101'. By appending these extracted bits, we obtain the secret message as '0110101'.

3. The proposed method

The techniques presented in Section 2 make distinction between smooth and sharp areas, however without computing the actual edges in an image. Chen et al. (2010) take advantage of a method for computing the actual edges in an image in order to embed more secret message bits to edge pixels. The edge detector used in this technique is hybrid edge detector. That is, the edges found by a fuzzy edge detector and the Canny edge detector are unified in order to find a larger set of edges. The Canny edge detector is a classic one found in image processing textbooks (Jain, Kasturi, & Schunck, 1995; Trucco & Verri, 1998). Fuzzy edge detectors are based on fuzzy logic. In recent years, many fuzzy techniques have been proposed in the literature for edge detection. In our method, a hybrid edge detector has also been used. For simplicity, though, instead of the Canny edge detector we have used the Sobel and the Laplacian filters (Rangarajan, 2011). A description of all edge detectors used in our method is given in the next subsection.

3.1. Preliminaries

The Sobel filter approximates the first derivative in a specific direction. It consists of two 3×3 convolution masks as given below:

Gx

−1	0	+1
−2	0	+2
−1	0	+1

Gy

+1	+2	+1
0	0	0
−1	−2	−1

The second mask equals the first mask rotated by 90 degrees. The gradient size for each pixel is given by the following formula:

$$|G| = \sqrt{Gx^2 + Gy^2}. \tag{19}$$

Typically, this quantity can be approximated by using the following formula

$$|G| = |Gx| + |Gy|. \tag{20}$$

The angle of the direction of the edge (in correspondence to the pixel grid) is given by

$$\theta = arctan(Gy/Gx). \tag{21}$$

The Laplacian filter is a 2-D isotropic measure of the second order derivative of an image. The Laplacian $L(x,y)$ of an image with pixel intensities $I(x,y)$ is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}. \tag{22}$$

Three usually used masks in order to compute the Laplacian are given below:

0	1	0
1	−4	1
0	1	0
1	1	1
1	−8	1
1	1	1
−1	2	−1
2	−4	2
−1	2	−1

The Laplacian of an image is given simply by a convolution with one of these masks.

As far as the fuzzy detector, the edge detection process is the following: Assume that the image X is an image of dimension $W \times H$, and all the pixels in X are gray level from 0 to L (i.e. $x_{mn} \in [0,L]$). First the membership grade value $_{mn}$ at position (m,n) is determined. The image X is first transformed into an array F of fuzzy singletons $\mu_{m,n} \in [0,1]$ with $m \in [1,W]$ and $n \in [1,H]$. The array F is a union of all μ_{mn}'s and is determined as

$$F = \bigcup_{m=1}^{W} \bigcup_{n=1}^{H} \mu_{mn}, \tag{23}$$

where, if x is the biggest grayscale value in X, $\mu_{mn} = \frac{x_{mn}}{x}$. The simplest way to define a fuzzy edge detector is the determination of a proper membership function $\bar{\mu}_{mn}$ for each pixel x_{mn} with a surrounding $w \times w$ spatial window

$$\bar{\mu}_{mn} = min\left(1, \frac{\tau}{w} \sum_i \sum_j min(\mu_{i,j}, 1 - \mu_{i,j})^{\rho}\right)^{\frac{1}{\rho}}, \tag{24}$$

where

$$\mu_{mn} = \frac{\{max(x_{ij}) - min(x_{ij}) : i,j \in [1,w]\}}{x}. \tag{25}$$

Suggested values of 'τ' and 'w' are 9 and 3, respectively.

Let us assume that F' is an image containing all edges of image F in the fuzzy domain. The image is also an array of fuzzy singleton $\bar{\mu}_{mn}$ and is determined as

$$F' = \bigcup_{m=1}^{M} \bigcup_{n=1}^{N} \bar{\mu}_{mn}. \tag{26}$$

In our technique we take the edges resulting from the fuzzy edge detector (Chen et al., 2010) ORed with the edges resulting from the Sobel edge detector resulting into the Sobel OR fuzzy edge image; or we take the edges resulting from the fuzzy edge detector ORed with the edges resulting from the Laplacian edge detector resulting into the Laplacian OR fuzzy edge image. An example of the edge detectors we implemented is shown in Fig. 1.

3.2. Description of the method

3.2.1. Choosing the appropriate bits for embedding

Finally, we show the high payload technique for color images by El-Emam (EL-Emam, 2007) which is related to our method. The technique is working for 24-bit bitmap images whose pixel consists of three bytes for the Red, Green and Blue colors. A main case is defined to have a value between 1 and 16. The formula by which the main case for a pixel color is computed is given by:

$$MC = Int\left(\frac{\text{ByteColor}}{16}\right) + 1, \tag{27}$$

where ByteColor $\in \{ByteRed, ByteGreen, ByteBlue\}$ corresponds to the value of the color in decimal form. In order to hide data of larger size all three colors of each pixel are checked for whether they are suitable to be used for information hiding based on sub-cases (SCs). 6 sub-cases were defined based on the following: Suppose that $MC^{color} = index$, where $1 \leqslant index \leqslant 16$ and $color \in \{R,G,B\}$, where R, G, B are the colors of the pixel equal to ByteRed, ByteGreen and ByteBlue, respectively. Suppose that the MC of the current pixel is C and X, Y are the MCs of the rest of the colors in the same pixel. Then, we can define the value of SC based on the following conditions:

$$SC = \begin{cases} 1 \Longleftrightarrow X, Y < C, \\ 2 \Longleftrightarrow (X = C \ \& \ Y < C) | (X < C \ \& \ Y = C), \\ 3 \Longleftrightarrow X, Y = C \\ 4 \Longleftrightarrow (X > C \ \& \ Y < C) | (X < C \ \& \ Y > C), \\ 5 \Longleftrightarrow (X > C \ \& \ Y = C) | (X = C \ \& \ Y > C), \\ 6 \Longleftrightarrow X, Y > C. \end{cases} \tag{28}$$

In the following table the 16 cases and the sub-cases they can include are shown:

MC	The corresponding *SC*	Set description of *SC*
1	$\{SC_3, SC_5, SC_6\}$	$\{\forall SC \ni C = Sel_{Color} \in CP \& RC_i \in CP$ $\forall\ i = 1, 2 \ni C \leqslant RC_i\}$
2–15	$\{SC_1, SC_3, SC_5, SC_6\}$	$\{\forall\ SC \ni C = Sel_{Color} \in CP \& RC_i \in CP$ $\forall\ i = 1, 2 \ni C \leqslant RC_i | C \geqslant RC_i\}$
16	$\{SC_1, SC_3\}$	$\{\forall\ SC \ni C = Sel_{Color} \in CP \& RC_i \in CP$ $\forall\ i = 1, 2 \ni C \geqslant RC_i\}$

where, CP (Current Pixel) is the current pixel that includes the set of colors R, G, B Sel_{Color} is the selected color for which it holds:

$$Sel_{Color} = argmin_{\forall Color \in CP}\{MC_{Color}\} \quad (29)$$

C_{mc} is the basic case of the selected color in the current pixel for which case it holds:

$$C_{mc} = MC_{Sel_{Color}} \ni Color \in CP \quad (30)$$

RC_i are the rest two colors of the pixel (the colors besides the selected one).

As it can be seen from the table, the sub-cases SC2 and SC4 are rejected from all the main cases. This is the case in order to avoid abrupt changes in the stego-image. The pseudo-code that shows how the bits that can be used to embed the secret message is given below:

```
If (Cmc>=2 AND SC = 1) {
        //Check the MC of rest colors
        If (RC = 1) {
                //Check if all current pixel property is equal to
    the next pixel property
                If (CP.property = NP.property) {
                    //Hide 2 data bits in the selected color
    from the current pixel
                    Hide (CP.C, DataBits (2));
                    //Hide 2 data bits in the selected color byte
    from the next pixel
                    Hide (NP.C, DataBits (2));
                }
            }
}
Else if (Cmc>=1 AND Cmc<=16 AND SC = 3) {
            //Hide 1 data bit in the Red color byte
            Hide (CP.R, DataBits (1));
            //Hide 1 data bit in the Green color byte
            Hide (CP.G, DataBits (1));
            //Hide 2 data bits in the Blue color byte
            Hide (CP.B, DataBits (2));
}
Else if (Cmc>=1 AND Cmc<=15 AND SC = 5) {
            //Hide 2 data bits in the selected color byte from
    the current pixel
            Hide (CP.C, DataBits (2));
            //Hide data bits in the greater or equal MC byte for
    the selected color min case from the
            //rest pixel bytes
            Hide (CP.E, DataBits (2));
}
Else if (Cmc>=1 AND CP.Cmc<=15 AND SC = 6) {
            //Hide 2 data bits in the selected color byte from
    the current pixel
            Hide (CP.C, DataBits (2));
            //Hide data bits in the greater or equal MC byte for
    the selected color min case from the
            //rest pixel bytes
            Hide (CP.H, DataBits (2)); }
```

3.2.2. Hiding the secret image

1. Initially the user selects the cover-image.
2. Next, the grayscale version of the image that the user has selected is computed by using the following formula for each pixel

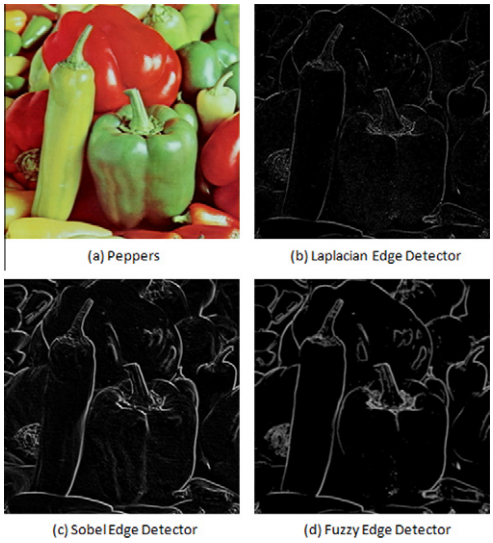

Fig. 1. Example of the edge images we generated.

$$I = 0.299 * R + 0.587 * G + 0.114 * B \quad (31)$$

where R is the value of the red channel of the specific pixel, G is the value of the green channel of the pixel and B is the value of the blue channel of the pixel.

3. Since the gray-scale image has been computed the edge detection filters are applied to it resulting to the Sobel OR fuzzy and the Laplacian OR fuzzy images.
4. After the edge detection process the method by EL-Emam (2007) is used to compute the number of embedded bits for each Red, Green and Blue channel of each pixel. In order to enhance the size of the embedded message, we examine whether each pixel belongs to an edge of the image. If this is the case, the number of bits that are to be embedded is increased by one. For example, if the selection method (EL-Emam, 2007) has resulted that in a pixel 2 bits in the red channel, 0 bits in the green channel and 2 bits in the blue channel can be changed and the pixel is an edge pixel, 3 bits in the red channel are changed, 1 bit in the green channel and 3 bits in the blue channel. This alternation is based on the fact that changes in edge pixels are more difficult to be detected. After this, the size of the secret message that the image can hide can be computed, based both on the Sobel OR Fuzzy and the Laplacian OR Fuzzy edge detectors. However, for security reasons, not all the pixels in the image are used but a pseudorandom number generator is employed, that computes the step with which the pixels to be used are determined. This generator extracts as its result one of the number 0, 1 or 2 and in the result we add the value 1 to conclude whether we move 1, 2 or 3 pixels forward. Substantially, in the best case we use 100% of the image capabilities, while in the worst case we use 30%.
5. If step 4 finishes the user is asked to select the image that constitutes the secret message. Of course, the size of this image must not outreach the maximum allowed size for the secret image computed in step 4.
6. When the user selects the secret message, this is embedded in the cover image and the two stego-images (corresponding to the Sobel OR fuzzy and the Laplacian OR fuzzy) are computed.
7. Finally, two auxiliary files that required in the extraction process of the secret message must be created. One file is created with respect to the stego-image that has resulted using the Sobel OR Fuzzy edges and one file is created with respect to

Fig. 2. Four 128 × 128 color images.

the stego-image that has resulted using the Laplacian or fuzzy edges. These files contain the width and the height of the image that constitutes the secret message as well as the elements of the table that contains the number of bits that can be changed in each color channel of each pixel of the cover image. This information is needed in order to read the correct bits in the reverse process in order to build the secret image. However, because this information is extremely important in revealing the secret message, it is not safe that this file is sent in the clear, since a third party can realize its importance and recover it. For this reason, these files are encrypted using the Triple-DES algorithm. The application asks the user to insert the secret key that is required for the encryption and then to confirm it. After the key is confirmed the encryption process starts, the two files are saved and the procedure finishes.

3.2.3. Extracting the secret message

1. Initially, the user loads the stego-image where the secret message that he/she wants to see is hidden. Also, the user inserts the name that he/she wants to use for the secret image.
2. Next, the user is asked to insert the secret key that was used for the encryption process and loads the encrypted file that was used besides the stego-image.
3. Since the user inserts all the above elements, the appropriate bits from the stego-image are read (according to the information of the file) and the secret image is extracted.

4. Experimental results

In this section, we present the results from the experiments we conducted in order to evaluate the performance of our proposed method. For convenience in making comparisons with the related scheme in Chen et al. (2010), the images used in the experiments of this paper are the same with those used in Chen et al. (2010) except for the fact that in our case are color images and not grayscale. All four images are shown in Fig. 2. In order to compare the two methods, the hiding capacity (in bits) divided by 3 (we use color images, while in Chen et al. (2010) grayscale) and the PSNR values are used. peak signal to noise ratio (PSNR) is a statistical measure commonly used in image steganography for indicating the quality difference between the stego-image and the cover image. PSNR is estimated (in decibel) by the following formula:

$$PSNR = 10log_{10}\left(\frac{255^2}{MSE}\right), \tag{32}$$

where MSE is the mean square error which is defined as:

$$MSE = \frac{1}{wh}\sum_{i=1}^{w}\sum_{j=1}^{h}(C_{ij} - S_{ij})^2, \tag{33}$$

where w and h are the width and height of the images and C_{ij}, S_{ij} are the value of the pixel (i,j) in the cover and stego image, respectively. Stego-images with higher PSNR are evaluated as better.

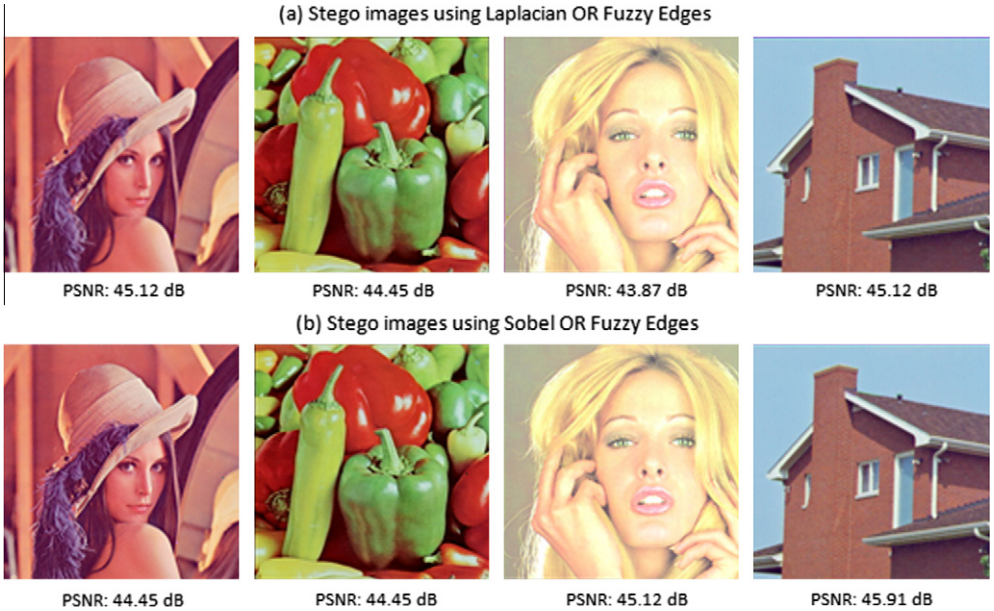

Fig. 3. Stego images generated by the proposed method.

Table 1
Comparison of results for the proposed method and the mehod in (Chen et al.).

Lena	Chen et al. method	Our method					
		Laplacian OR fuzzy edges			Sobel OR fuzzy edges		
		RNG with step 1,2,3	RNG with step 1,2	NO RNG used	RNG with step 1,2,3	RNG with step 1,2	NO RNG used
Capacity	10,662 (bits) 0.65 (bpp)	15,295 (bits) 0.93 (bpp)	20,596 (bits) 1.26 (bpp)	30,987 (bits) 1.89 (bpp)	15,213 (bits) 0.93 (bpp)	20,490 (bits) 1.25 (bpp)	30,811 (bits) 1.88 (bpp)
PSNR	47.1	46.88	46.88	45.12	45.91	46.88	44.45

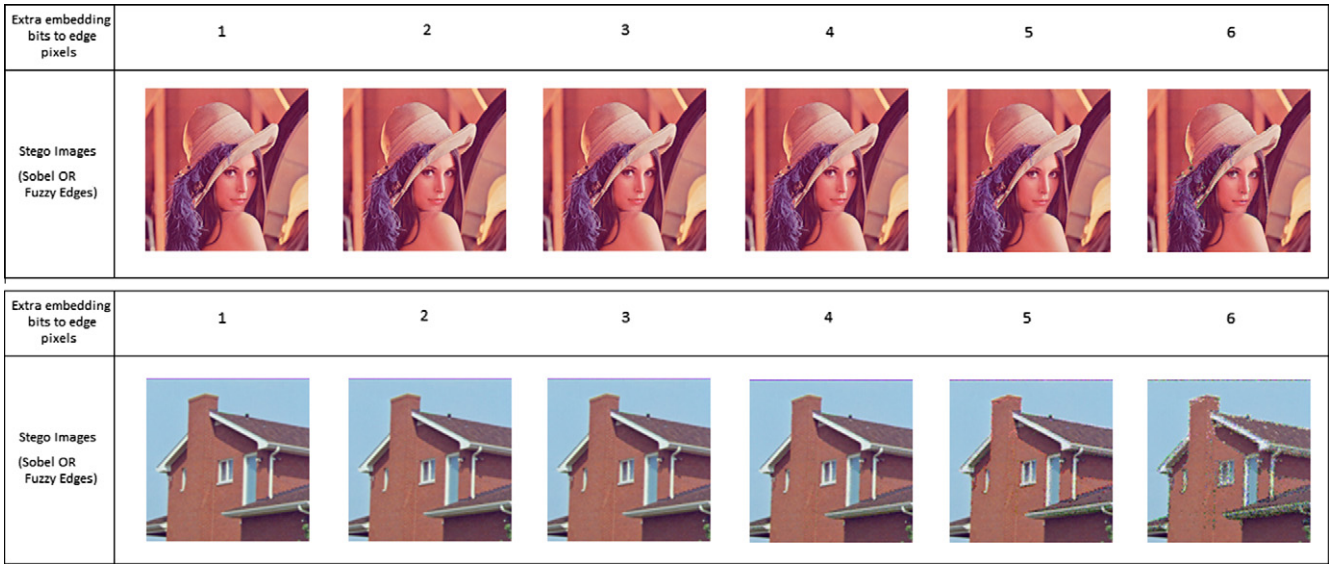

Fig. 4. Stego images resulted when the edge pixels bits for embedding are increased by 1 to 6.

Fig. 3 shows the quality of the stego images that are generated by the proposed method. It can be seen that the stego-images are visually indistinguishable from their corresponding cover images and the PSNR value is quite high for all images. As described in previous section, a random number generator (RNG) with step 1, 2 or 3 is used in our method for security reasons. In the experiments we conducted, though, we also tested our method with a RNG with step 1, 2 and with no RNG at all. According to the results shown in Table 1, the embedding capacity is significantly increased by our method in relation with the method proposed in Chen et al. (2010), while the image quality (PSNR) is almost the same. In addition, when no random number generator is used in our method, the PSNR, as expected, is slightly reduced but the capacity in bits is doubled. Specifically, for the "Lena" image, when this option is selected and therefore bits of the stego-image are included in almost every pixel of the image, PSNR almost reaches the value of 45, while the technique by Chen et al. (2010) achieves a PSNR of 47.1. However, in our case the number of bits per pixel included almost reaches the value of 2, while the technique by Chen et al. (2010) performs poorly achieving the value of 0.61 bpp.

From Fig. 3 it is obvious that the stego-images are not significantly distorted, independently of whether we are using the "Laplacian or fuzzy" or the "Sobel or fuzzy" edge detector.

5. Gradual increase in the number of bits hidden in edge pixels

Except for the above experiments, we also wanted to investigate the resistance of the proposed steganographic method in a gradual increase of the bits that are used for the embedding of the secret image in the edge pixels. As already mentioned, our method increases by one the number of the bits which can be used for embedding in each color band, if a particular pixel belongs to an edge. The highest number of bits obtained from the algorithm we use in order to select the pixels and consequently the bits in which the secret message can be embedded is 2. In the experiment we conducted we increased the embedding bits in the edge pixels in a range from 1 to 6 bits for every RGB channel. The stego images obtained from the experiment for the cover images Lena and Building are shown in Fig. 4. As we can see, the image quality remains good even when we use 3 extra bits for embedding in the edge pixels. As expected, the quality of the images reduces significantly when we use 4, 5 and 6 additional bits.

6. Conclusions and future work

In this paper, we presented a simple technique merging a high payload method (Chen et al., 2010) and a method that includes stego bits in edge pixels derived by a hybrid edge detector (Chen et al., 2010). The idea of our approach is simple. We compute the number of bits to be embedded in a pixel, based on the high payload technique and include one more bit for each RGB channel if the current pixel examined is an edge pixel as extracted by the hybrid edge detector.

As one can see from the experimental results, our method outperforms the hybrid edge detector technique which is considered as state of the art.

However, further improvements to the presented algorithm can be made if the relations between neighboring pixels are taken into account, an issue that is not examined by the presented method.

References

Chang, Chin-Chen, & Tseng, Hsien-Wen (2004). A steganographic method for digital images using side match. *Pattern Recognition Letters, 25*(12), 1431–1437.

Chen, Wen-Jan, Chang, Chin-Chen, & Hoang Ngan Le, T. (2010). High payload steganography mechanism using hybrid edge detector. *Expert Systems with Applications, 37*(4), 3292–3301.

EL-Emam, Nameer N. (2007). Hiding a large amount of data with high security using steganography algorithm. *Journal of Computer Science, 3*(4), 223–232.

Fridrich, J., Goljan, M., & Du, R. (2001). Reliable detection of LSB steganography in color and grayscale images. *IEEE Multimedia, 8*, 22–28.

Jain, R., Kasturi, R., & Schunck, B. G. (1995). *Machine vision*. McGraw-Hill.

Kim, T. (1992). Side match and overlap match vector quantizers for images. *IEEE Transactions on Image Processing, 1*, 170–185.

Morkel, T., Eloff, J. H. P., Olivier, M. S. (2005). An overview of image steganography. In *ISSA'05* (pP. 1–11).

Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1999). Information hiding – A survey. *Proceedings of the IEEE, 87*, 1062–1078.

Queirolo, F. (2011). Steganography in images. Accessed 09.11.

Rangarajan, S. (2011). Steganography in images. Accessed 09.11.

Thien, C.-C., & Lin, J.-C. (2003). A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function. *Pattern Recognition, 36*(12), 2875–2881.

Trucco, A., & Verri, E. (1998). Introductory Techniques for 3-D Computer Vision. Prentice Hall.

Wang, C.-M., Wu, N.-I., Tsai, C.-S., & Hwang, M.-S. (2008). A high quality steganographic method with pixel-value differencing and modulus function. *Journal of Systems Software, 81*, 150–158.

Wu, D.-C., & Tsai, W.-H. (2003). A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters, 24*(910), 1613–1626.