# An algorithm to hide information in binary images

JOAN-JOSEP CLIMENT, JAVIER SANTACRUZ, LEANDRO TORTOSA, ANTONIO ZAMORA
Departament de Ciència de la Computació i Intel·ligència Artificial
Universitat d'Alacant
Ap. 99 E-03080 Alacant
SPAIN

*Abstract:* The objective of this paper is to develop a method to hide information inside a binary image. An algorithm to embed data in scanned text or figures is proposed, based on the detection of suitable pixels, which verify some conditions in order to be not detected. In broad terms, the algorithm locates those pixels placed at the contours of the figures or in those areas where some scattering of the two colors can be found. The hidden information is independent from the values of the pixels where this information is embedded. Notice that, depending on the sequence of bits to be hidden, around half of the used pixels to keep bits of data will not be modified. The other basic characteristic of the proposed scheme is that it is necessary to take into consideration the bits that are modified, in order to perform the recovering process of the information, which consists on recovering the sequence of bits placed in the proper positions. An application to banking sector is proposed for hidding some information in signatures.

*Key-words:* Data hiding, watermark, steganography, binary images, image processing, banking security.

## 1 Introduction

Data hiding, a form of steganography, is the process of encoding extra information in an image by making small modifications to its pixels. So, by means of some algorithm, we are able to embed data into digital images for the purpose of identification, annotation, copyright, and others. It is important to note that the hidden data must be perceptually invisible yet robust to common signal processing operations.

Depending on what information in which form is hidden in the image, one can distinguish at least two types of data hiding schemes: non-robust and robust image watermarking. In the first one, a digital image serves as a container for a secret message. For example, by replacing the least significant bit of each pixel (LBS) with an encrypted bit-stream, the changes to a typical image will be imperceptible and the encrypted message can be masked by some innocent looking image. In the robust case, a short message (a watermark) is embedded in the image in such a way that image is able to survive common image processing operations, such as loosy compression, filtering, noise adding, geometrical transformations, and so on.

Data hiding is closely related to cryptography. In broad terms, the purpose of cryptography is to keep messages unintelligible to those who not posses the proper keys to recover them. But, sometimes, it may be desirable to achieve security and privacy by masking the presence of communication. This is the main goal addressed by steganography, as usually is named the data hiding.

We can say that each particular data hiding scheme consists mainly of an embedded algorithm and a detector function. The embedding algorithm is protected by a key-word with the aim that only those who knows this secret key can access the information.

Classifications and surveys of information hiding can be found in [2, 1], while a review to apply theses techniques in image, audio, and text is in [3]. When we try to hide information in two-color (black and white) images, (like facsimiles, xeroxs and bar codes) we find hard problems because the change of a pixel in images like those can be easily detectable. Nevertheless, some schemes have been proposed for this task. We remark the steganography scheme proposed by Yu-Yuan Chen, Hsiang-Kuang Pan, and Yu-Chee Tseng [4], which ensures that in each $m \times n$ image block of the host image, as many as $\lfloor \log_2(mn+1) \rfloor$ bits can be hidden in the block by changing at most two bits in the block. Some other examples have been proposed, based on the idea of manipulating some pixels according to some rules, playing an important role the characteristics of the neighborhood pixels, (see [5, 6]).

## 2 Some basic digital image definitions

A digital image $a[m, n]$ described in a 2D discrete space is derived from an analog image $a(x, y)$ in a 2D continuous space through a sampling process known as digitization, which basically consists on dividing the original image $a(x, y)$ in $M$ rows and $N$ columns. A pixel is termed as the intersection of a row and a column. The value assigned to the integer coordinates $[m, n]$, with $m = 0, 1, \ldots, M - 1$ and $n = 0, 1, \ldots, N - 1$ is $a[m, n]$.

There are standard values for the parameters $M$, $N$ and $L$, the gray level. Quite frequently, we see cases of $M = N = 2^k$, where $k = 8, 9, 10$. The number of gray levels is usually a power of two, that is, $L = 2^B$, where $B$ is the number of bits in the binary representation of the brightness levels. When $B \neq 1$, we speak of a gray level image. If $B = 1$ we speak of a binary image. These are the images we are going to deal with along this paper, images with two gray levels, black and white, or $0$ and $1$.

In this section we are not going into further details about the theoretical properties of digital images, but we must point out some fundamental definitions about operations with digital images. The types of operations that can be applied to digital images to transform an input image $a[m, n]$ into an output image $b[m, n]$ can be classified into three categories:

1. Point: the output value for a coordinate is dependent only on the input value at the same coordinate.

2. Local: the output value for a coordinate is dependent on the input values in the neighborhood of that coordinate.

3. Global: the output value for a coordinate is dependent on all the values in the input image.

Due to the fact that the generic complexity (per pixel) of the global operations is $N^2$, if the image size is $N \times N$, we are interested in performing point or local operations, which complexity is much less. When local operations are performed to process an image, various neighborhoods can be used. The most common used neighborhood is the rectangular sampling, where images are sampled by laying a rectangular grid over the image. A pixel $p$ at coordinates $(x, y)$ has four horizontal and vertical neighbors, whose coordinates are given by

$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1).$$

This set can be denoted by $N_4(p)$. Similarly, the four diagonal neighbors of $p$ have coordinates

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

and are denoted by $N_d(p)$. The set $N_4(p) \cup N_d(p)$ are called the 8-neighbors of $p$, denoted by $N_8(p)$. In a similar way, we can determine the 16-neighbors of $p$, denoted by $N_{16}(p)$. The sets $N_8(p)$ and $N_{16}(p)$ will play an important role in our algorithm.

The two basic ways to embed data in binary image are by changing the values of individual pixels and by changing a group of pixels. The first approach flips a black pixel to white or vice versa. The second approach modifies such features as the thickness of strokes, curvature, relative positions, etc. In this paper, we focus on using the first approach.

## 3 An algorithm to embed information in digital binary images

### 3.1 Description of the algorithm

The idea in which our algorithm is based is to locate certain pixels of the host image, which must fulfill some special requirements that we will analyze in detail in this section. In broad terms, we are interested in finding pixels located in the contours of the figures or in those areas where we can find certain scattering of pixels with the two colors (black and white). The hidden information is independent from the values of the pixels where this information is going to be embedded. Therefore, we expect that there will be the half of coincidences between the bit we want to hide and the value of the pixel in the host image. We must point out that no information is hidden in the solid areas of the figure. The reason is simple: we want to have the control on the quality of the image after modification and modifying such pixels in solid areas can be easily detected by an observer.

The host image is represented by a matrix whose entries are 0s and 1s, where $0$ represents a black colored pixel and $1$ represents a white colored one. The information or data we want to hide is not but a sequence of bits. No kind of separation or assembling is distinguished, so the chosen alphabet has no influence on the algorithm. To begin with, the algorithm works with natural chains of $8$ bits, that is, an ASCII character; nevertheless, a previous function can be executed to substitute each character of the used alphabet by the binary representation. Moreover, the algorithm takes into account a sequence which we are going to identify with the end of the process of hiding data. That special sequence is composed by a byte of 1s. It is important to notice that we analyse the image processing it by rows, verifying that each pixel assembles the proper conditions to keep a bit of information in it. So, we will perform the computations over each pixel individually.

Given two pixels $p = (x, y)$ and $q = (s, t)$, we define the $D_8$-distance between $p$ and $q$ as

$$D_8(p,q) = \max\left(|x-s|, |y-t|\right). \tag{1}$$

From equation (1), we see that pixels with $D_8$-distance from $(x,y)$ less than or equal to some value $r$ form a square centered at $(x,y)$. For example, the pixels with $D_8$-distance $\leq 2$ from $(x,y)$ form the following contours of constant distance

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array} \tag{2}$$

As we can see in matrix (2), the pixels with $D_8$-distance equals 1 form the set $N_8(p)$, while the pixels with $D_8$-distance equals to 2 form the set $N_{16}(p)$. We call $N_8(p)$ and $N_{16}(p)$ the *inner bound* and the *outer bound* of $p$, respectively.

Now, we describe the whole process leading to embed information in a host image. The algorithm analyses, by rows, each pixel of the matrix representing the host image, to check if it is possible to keep a bit of the sequence originated by the secret information. This process to detect those pixels which are optimum to keep information, can be resumed in the following steps:

*Step 1.* Determine if the pixel $p = (x,y)$ verifies the proper conditions to hide information.

*Step 2.* Check if pixel $p$ is placed under the influence of other pixels where information has been previously hidden.

*Step 3.* Check if, when keeping information in pixel $p$, the conditions of the neighboring pixels have been modified.

In step 1, we impose two necessary, but not sufficient, conditions to determine if the pixel $p$ is suitable or not to embed information in it. These conditions can be expressed as:

*Condition 1.* Given the pixel $p$ at coordinates $(x,y)$, pixels in the set $N_8(p)$ must have the same value as the pixel placed at coordinates $(x-1, y-1)$.

*Condition 2.* Consider the sequence given by the values of pixels in $N_{16}(p)$. This sequence must contain the value of the pixel $(x-2, y-2)$ between $5$ and $12$ times.

These are the two conditions that a pixel must verify as a first step to go on with the process. Remark that the algorithm begins to analyse the pixels from the position $(4,5)$ in the matrix of the host image.

In step 2, given that the pixel $p$ verifies condition 1 and condition 2, we have to check that $p$ is not above the influence of pixels where some information was hidden before. What we mean is that we cannot hide information in the pixel $p$, if it is placed in the sets $N_8(p')$ or $N_{16}(p')$, for a neighbor pixel $p'$ where we have previously embedded a bit of information.

We solve this problem by means of an auxiliary matrix, with three rows and so many columns as horizontal pixels the image has. In this matrix, we place a value $1$ in those positions where a bit of information exists, and a value $0$ in other case. The third row in this matrix corresponds with the row in the image where the pixel we are considering is placed. The first and second rows are the ones where we can find the pixels that interfere with the one we are dealing with. So, the pixels that can produce problems are those located along the inner and outer border; more exactly, those located in the upper side and left side. If we find in some of these positions a value of 1, the pixel we are considering must be rejected, and we go to the next one. When a row of the image has been completed, the rows of the auxiliary matrix are shifted one position upwards, inserting then a null row.

In step 3, we can not affirm that the pixel $p$ is suitable to hide information if we do not check that, in the case to be modified, do not change the conditions of the previous pixels studied. May be possible that when a bit is modified to hide a bit of information, this can originate a deep change in the conditions for the inner and outer border of other neighbor pixels. This fact can produce that a pixel which was not suitable to hide information before the modification of the pixel $p$, will became a suitable pixel to embed information. A situation like this is very dangerous because it produces confusion in the process of recovering the data. Consequently, this pixel must be rejected. To avoid this situation, we have to analyse all the pixels located at the upper and left borders. More exactly, we refer to the pixels at coordinates $(x-2, y-2)$, $(x-1, y+2)$ and $(x, y-2)$, $(x, y-1)$. So, the task we must perform with these pixels is the study of their behavior when we place $0$ and $1$, respectively, as the value of the pixel $p$. If any of the previous pixels transforms to a suitable pixel to hide information, we do not take into account the pixel $p$, and continue with the following one, which may be that located at position $(x, y+3)$.

Once a pixel has overcome the conditions imposed through the three steps of the process described above, we can hide a bit of information in that position. Note that the algorithm works with individual pixels; after the process is completed for a concrete one, we go to the next position and begin again with the three steps.

One of the main characteristics in the above scheme is that, although many pixels are not modified to hide information, the bits that are modified are not selected from a random process. We always take into account the positions where the bits are modified and it is taken into consideration. As a consequence of this, the process to recover the hidden information coincides exactly with the process to embed information. The only thing we have to remark is that, in this algorithm, we only have to recover the sequence of bits placed in the proper positions. Now, we do not have to modify any pixel. The process ends when we find the character who finish the information (a byte of 1s).

## 3.2 Ax example with a concrete image and experimental results

The essential part of the algorithm consists on the election of those pixels where we can keep bits of information. With the purpose of understanding the way in that this process is carried out, we consider an example. We use a binary image, called *garra.bmp*, that represents the claw of a bird and whose dimensions are $57 \times 88$, (see Figure 1).

Figure 1 shows us a $13 \times 22$ matrix, whose entries are 0s and 1s representing a piece of the left image.

We use the matrix shown in Figure 1 to make some considerations about the hiding information process, developing the three steps described in Section 3.1. We can find some pixels verifying the conditions 1 and 2 in step 1. For example, pixels $p_1 = (7, 16)$ and $p_2 = (10, 10)$ verify these conditions. Pixel $p_1$ has the following characteristics: the value of all the pixels in the set $N_8(p_1)$ is equal to 0 and the set $N_{16}(p_1)$ has 6 pixels whose value is equal to 1 (the value of the pixel $(5, 14)$). Therefore, this pixel is, by now, a suitable one to keep information in it. Similarly, pixel $p_2 = (10, 10)$ is agree with condition 1 and 2. In this case, the set $N_{16}(p_2)$ has 8 pixels whose value is equal to 0 (the value of the pixel $(8, 8)$). It also verifies the step 1 of the algorithm.

Observing the matrix we are working with, note that the coordinates $(7, 16)$ and $(10, 10)$ are not under the influence of suitable pixels to keep information. Consequently, $p_1$ and $p_2$ are suitable to keep information, according to step 2. Now, we only have to check that they do not modify the conditions of the pixels in their influence area.

From matrix in Figure 1 we observe that when the pixel $(7, 16)$ takes the value $0$, the conditions of the step 1 are not given for the pixels in the influence area; nevertheless, when the value of this pixel changes to 1, we see that a deep change occurs in the conditions of the pixel placed at position $(5, 16)$, as we reflect in Figure 2. That is, pixel $(5, 16)$ becomes a suitable one to embed information in it. This fact may produce a problem when the information must be recovered. Consequently, pixel $(7, 16)$ must be rejected. This situation do not happen when we study the pixel $(10, 10)$. We observe that this one does not modify the conditions in the pixels located at the neighboring. In such case, we can keep a bit of information in that coordinate and this will not be detectable.

We have implemented our scheme and conducted some tests. The simulation is based on the generation of random matrices, which represent possible host images, used to embed some short information. We define three matrices, denoted by $M_1$, $M_2$ and $M_3$, whose sizes are $200 \times 200$, $300 \times 300$ and $450 \times 436$, respectively. The texts we will try to hide in matrices $M_1$, $M_2$ and $M_3$, are *Customer Number*, *Your Serial Number* and *Royal Festival*, respectively. Finally, we choose to repeat the generation of random matrices for five times.

The experimental results obtained for this simulation with these parameters are given in Table 1. The first column represents the type of matrix randomly generated. The second column shows us the total number of bits that we can keep in the generated matrix. The third column give us the number of coincidences between the values of the suitable bits to keep information in the matrix and the bit of information kept in that pixel. In other words, this column represents the number of pixels in the matrix that are not modified by the hiding information scheme. The last column give us the execution time, in seconds.

We must point out that in all the cases, we achieve the objective to hide the chosen text in the corresponding matrix. The capacity provides us the limit of the bits we are able to hide. Some modifications to the algorithm may be easily introduced to increase the capacity of hiding information. Finally, we want to remark the velocity of the algorithm to check the characteristics of the pixels representing the host image.

## 3.3 An application to handwriting signatures

In summary, we have presented a technique for hidding data in binary images. We can apply the algorithm exposed in this paper to the particular case of handwriting signatures, with the aim of keeping some information in the image of the digitalized signature. Nowadays, banks or some other bussiness need to have a database with the signatures of their customers.

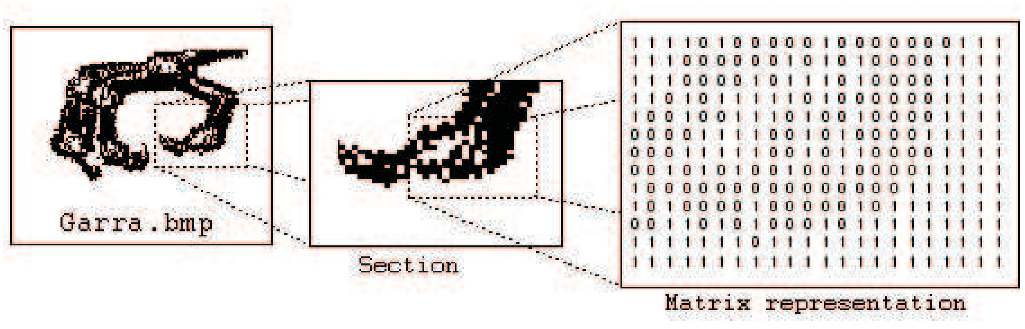

Figure 1: Example image.

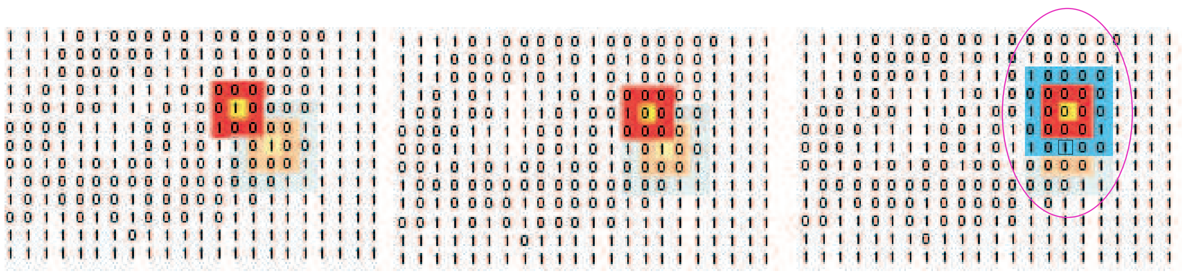

Figure 2: Influence area of pixel $(7, 16)$ when it is equal to $1$.

Table 1: Simulation of hiding information scheme described in this paper.

| Matrix | Capacity | Coincidence | Execution Time |
|---|---|---|---|
| $M_1$ | 210 | 71 | <1 |
| $M_2$ | 475 | 77 | <1 |
| $M_3$ | 1050 | 71 | <1 |
| $M_1$ | 177 | 60 | <1 |
| $M_2$ | 449 | 76 | <1 |
| $M_3$ | 1048 | 60 | <1 |
| $M_1$ | 174 | 58 | <1 |
| $M_2$ | 443 | 74 | <1 |
| $M_3$ | 991 | 56 | <1 |
| $M_1$ | 207 | 56 | <1 |
| $M_2$ | 509 | 84 | <1 |
| $M_3$ | 986 | 61 | <1 |
| $M_1$ | 190 | 66 | <1 |
| $M_2$ | 461 | 81 | <1 |
| $M_3$ | 930 | 60 | <1 |

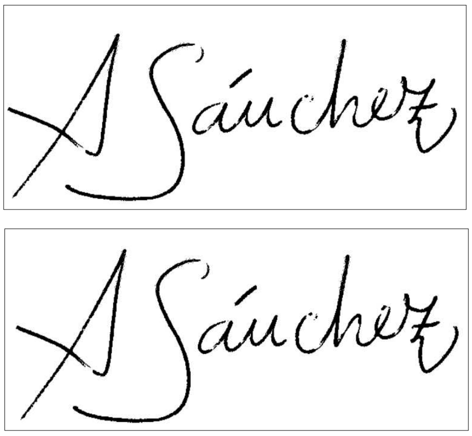

Figure 3: Digitalized signature.

We can introduce in the image some personal data, which may be convenient to know for us, but may be blind for the other part.

We see in Figure 3 an example of a digitalized signature, where we are going to embed some information. More exactly, the inserted text in the image is `25487996D`, which constitute the identification number for a particular person. With this technique, the total number of bits that we could keep in the whole image is $512$. The first character to introduce is $2 = (00110010)$. The eight bits of this byte are storaged in the coordinates

$$(20,179), (22,174), (24,182), (25,179),$$

$$(28,516), (29,181), (30,178), (30,520).$$

We can hide the rest of bits in the image following the algorithm described throughout this paper, resulting perceptually invisible, as we see when comparing both images in Figure 3. Image placed up represents the original digitalized signature, while image placed down represents the image after embedding the identification number.

## 4 Conclusions

In this paper, we have proposed a new data hiding scheme for 2-color images, based on the idea of locating some pixels placed at the contours of the figures or in those areas where some scattering of the two colors can be found. These suitable pixels are replaced, by the appropriate bit of information, from the sequence of bits we generate from the information that must be hidden. This means that not all the pixels where we keep data, have to be modified, what produces that the hiding effect is quite invisible. Experimental results show us that around half of the pixels need not be modified. Future research may be directed toward increasing the data hiding capacity, as well as reducing at most the visibility of the hiding effect.

*References:*

[1] Petitcolas, F.A.P., Anderson, R.J., Kuhn, M.: Information Hiding – A Survey. Proceedings of the IEEE (1999) 1062–78

[2] Anderson, R.J., Petitcolas, F.A.P.: On the Limits of Steganography. IEEE J. on Selected Areas in Communications **16(4)** (1998) 474–481

[3] Bender,W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for Data Hiding. IBM System **35(3-4)** (1996) 313–336

[4] Chen, Yu-Yuan, Pan, Hsiang-Kuang, Tseng, Yu-Chee: A Secure Data Hiding Scheme for Two-Color Images. IEEE Symp. on Computers and Communications (2000)

[5] Wang, Hsi-Chun A.: Data Hiding Techniques for Printed Binary Images. Proceedings of the International Conference on Information Technology: Coding and Computing Las Vegas (2001) 55–60

[6] Wu, M., Tang, E., Liu, B.: Data Hiding in Digital Binary Image. IEEE Inter. Conference on Multimedia and Expo New York City **1** (2000) 393–396