Journal scientific and applied research, vol. 21, 2021
International Journal

Original Contribution

ISSN 1314-6289

ANALYSIS OF A STEGANOGRAPHIC METHOD FOR HIDING ONE IMAGE IN ANOTHER IMAGE USING MATLAB AND IMAGE PROCESSING TOOLBOX

Daniel R. Denev

KONSTANTIN PRESLAVSKI UNIVERSITY OF SHUMEN, 115 UNIVERSITETSKA, SHUMEN 9700,
E-mail: slimshady33@abv.bg

ABSTRACT: *Analysis of a Steganographic Method for Hiding an Image into a Cover Image Using MATLAB and Image Processing Toolbox: The paper presents an algorithm for hiding a 24-bit color image in bmp format in another 24-bit color image also in bmp format. The algorithm is based on the least-significant bit method and is implemented using MATLAB and Image Processing Toolbox. This algorithm can be applied for 100% reconstruction of the hidden image when four cover images are used*

KEY WORDS: *Cover image, Hidden image, LSB, Steganography*

1. Introduction

In recent years, cryptography has successfully developed the field of steganography (from the city of "steganos" - hidden, covered and "graph" - drawing or writing), dealing with encryption and insertion of messages in unattractive image and sound, with the goal is to hide the very fact of transmitting information and using cryptography.

In the late 1990s, several types of steganography were identified:

— classical;

— computers;

— digital steganography.

Classical steganography dates back to the 5th century BC. The ancient Romans, for example, wrote between the lines with ink made from natural ingredients, such as fruit juices, milk, etc., transmitting information through wooden objects in which hidden information is is protected by a thin layer of wax.

Computer steganography is based on two principles. The first principle is that files containing digital images or sound can be modified without compromising their functions. The other principle is related to the inability of a person to distinguish minimal differences in the color of an image or in the quality of the sound, which allows the use of redundant information in these files. There are several ways to hide information in an image. One way is to hide it in the headers and footers of the images. The other way is to attach the data to the file, but its size can betray the fact of hiding information. The most common technology is the change of the least significant bit (LSB) in each pixel. Because this procedure manipulates existing information rather than adding new information, the file size does not increase, and the change is invisible to humans.

One of the most promising commercial areas of steganography is digital watermarks. The creation of invisible watermarks is used to protect the copyright of graphic and audio files. The watermarks included in the file can be recognized only by special programs that extract from the file information about the time of its creation, who owns the copyright and how to contact the author [1], [4].

2. Exposure

Images typically use 8-bit or 24-bit color. In 8-bit mode, up to 256 colors are defined, forming the image palette, and each color is denoted by an 8-bit value. 24-bit mode uses 24 bits per pixel and provides a much better set of colors. In this case, each pixel is represented by three bytes, each byte representing the intensity of one of the three primary colors - red, green and blue (red, green, blue - RGB).

The size of the image file is directly related to the number of pixels and the number of bits to represent the color. 640 x 480 photos using a 256-color palette require a 307 KB file. High resolution photos, 1024 x 768, with a 24-bit color image will result in a 2.36 MB file. To avoid sending files of this huge size, various types of compression have been developed, such as BMP, GIF and JPEG files.

GIF and 8-bit BMP files use lossless compression, which allows the software to accurately restore the original image. The JPEG format uses lossy compression, which means that the compressed image is almost the same as the original, but not exactly a duplicate. Although JPEG can also be used for steganographic applications, it is more common to hide data in GIF or BMP files. Several methods are used to hide information in an image, audio or video file, the most common of which is the LSB (Least Significant Bit) method. If 24-bit color is used, the change will be minimal and invisible to the human eye. Three adjacent pixels (nine bytes) with RGB encoding are considered as an example:

10010101 00001101 11001001 | 10010110 00001111 11001010 | 10011111 00010000 11001011

The goal is to "hide" the following nine bits of 101101101 data, with the hiding data usually compressed before the hiding process. If these nine bits are written on the youngest bits of the nine bytes above, the following sequence is obtained, with only the darker bits changed:

10010101 00001100 11001001 | 10010111 00001110 11001011 | 10011111 00010000 11001011

It should be noted that nine bits were successfully hidden, changing only four, or approximately 50% of the number of least significant bits.

Similar methods can be applied to 8-bit color, but the changes will be much more noticeable to the human eye. Most steganography experts recommend the use of images presented through 256 levels of gray, which provides better concealment of information [2], [5], [6].

The article examines an algorithm for hiding one image (hidden, F16 fighter) in another image (COVER, sunrise in San Clemente) and retrieving the hidden image, using the MATLAB software environment and the Image Processing Toolbox extension. The two images must be the same size, which in this case is 310 x 516, after artificially "cutting" the COVER image to the required size. The processed images are of the BMP type with 24-bit color depth.

The algorithm for hiding the image hidden in the COVER image is illustrated in Fig. 1. Each pixel (I, J) of the COVER image is represented by 8-bit sequences of red, green and blue color components, the bits of which are marked with R1… R8, G1… G8 and B1… B8 respectively. Each pixel (I, J) of the hidden image is also represented by 8-bit sequences of the red (r1… r8), green (g1… g8), and blue (b1… b8) color components.

The hiding process contains the following steps:

The hidden array stores the color 24-bit image of the F16 fighter to be hidden, and the COVER array stores the color 24-bit image of sunrise in San Clemente, which will be used to transfer the hidden image. Both arrays are 310 x 516 x 3 (three-dimensional for color images) and contain 8-bit unsigned integers.

Select the number n, $1 \leq n \leq 7$, reflecting the number of the smallest bits of the COVER image that will be used to hide the oldest bits of the hidden image.

The variable BC is formed, which contains the result of the execution of the bitwise complement operation bitcmp of the variable $Pn = 2n - 1, n = 1 \div 8$ to 8-bit unsigned integer, i.e. as a result of the bitcmp operation, an 8-bit sequence of units is obtained, in which the last n units are replaced by zeros (Table 1). Values of n greater than or equal to 9 are not applicable in this case, as more than 8 digits are required to represent the number Pn in a binary number system.

Table 1. Execute the bitcmp operation

n	1	2	3	4	5	6	7	8	9
Pn = 2 n - 1	1	3	7	15	31	63	127	255	511
BC - dec	254	252	248	240	224	192	128	0	-
BC - bin	11111110	11111100	11111000	11110000	11100000	11000000	10000000	00000000	-

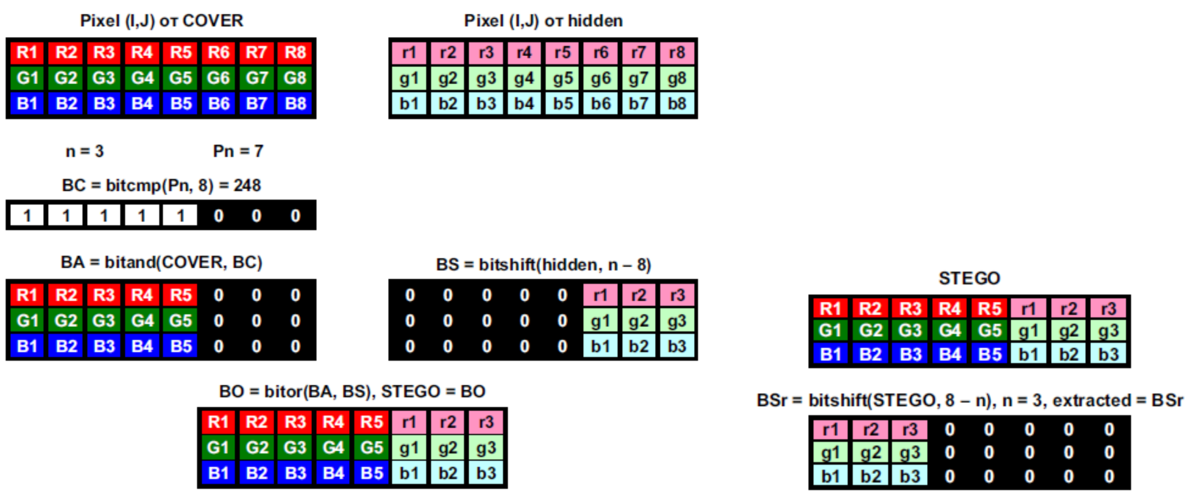

Fig. 1. Algorithm for hiding one image in another and extracting the hidden image from the stego-image

Reset the least n bits of the three-color components for each pixel of the COVER image by performing the bitwise multiplication operation bitand of the operands COVER and BC, which forms the variable BA [2], [3].

Reset the highest 8-n bits of the three color components for each pixel of the hidden image by performing the bitwise bitshift operation of the hidden operand on n 8 bits, i.e. at 8-n bits to the right. As a result, the variable BS is formed.

Formation of the variable BO when performing the operation bitwise logical addition bitor of the operand's BA and BS. The variable BO represents the stego-image, i. STEGO = BO. The process of retrieving the hidden image from the stego-image involves performing the bit-by-bit logical operation of the STEGO operand with 8-n bits to the left. In the extraction process, the youngest 8-n bits of the hidden image are lost, at which the quality of the recovered image is no longer the same.

3. Results

The results in the execution of the developed script are illustrated in the figures below. In Fig. 2a and Fig. 2b show the original images – sunrise (COVER) and fighter F16 (hidden). In Fig. 3 to Fig. 6 shows the corresponding STEGO i $(i = 1 \div 4)$ with the hidden image and the images extracted by them extracted i, using the least significant i bits of each of the color components (R, G, B) for each pixel of the COVER image.

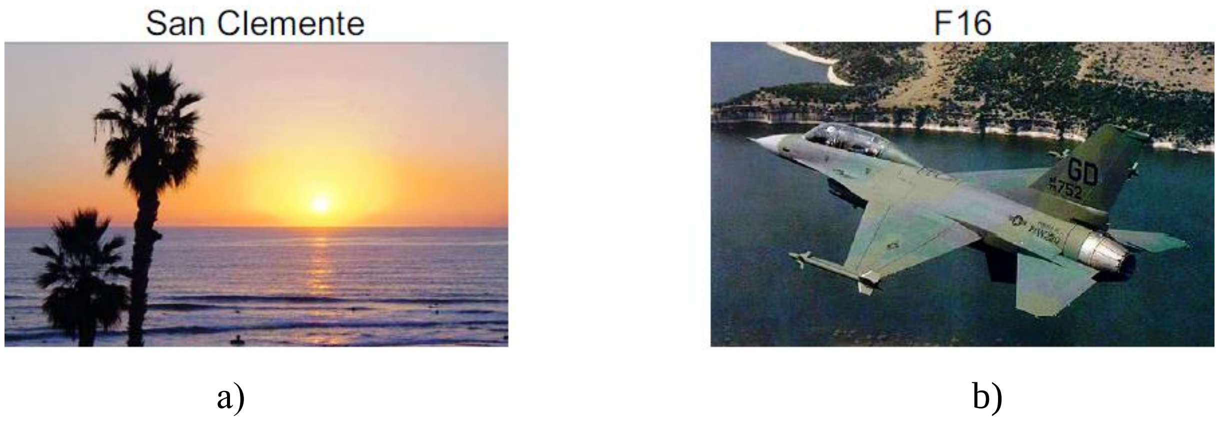

Fig. 2. Original images - sunrise in San Clemente (COVER) and fighter F16 (hidden)

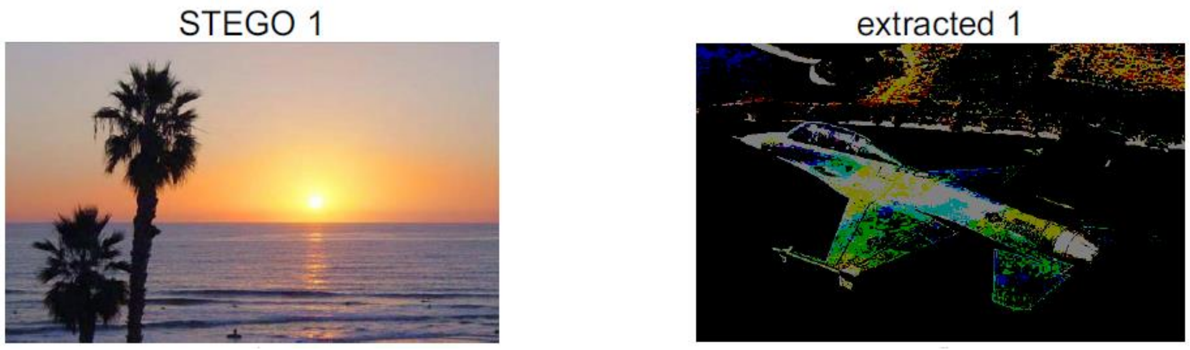

Fig. 3. Steganographic image (STEGO 1) with hidden information and the extracted image (extracted 1) using the lowest bit of the image COVER

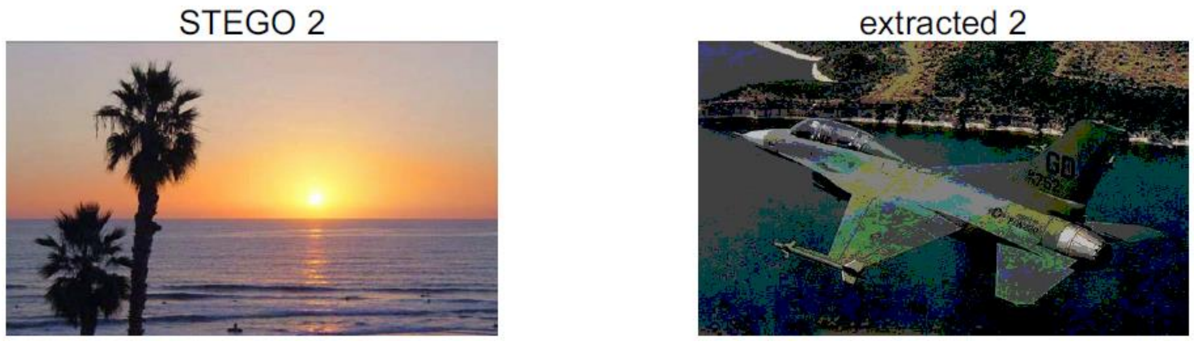

Fig. 4. Steganographic image (STEGO 2) with hidden information and the extracted image (extracted 2) using the two smallest bits of the image COVER

It can be seen that when using a maximum of three of the youngest bits in the COVER image to hide the hidden image, the hiding process is still not noticeable, but the recovered image is not of good quality, especially when using only the youngest bit. When using four, five, six or seven bits of the youngest bits in the COVER image, the hiding of information no longer goes unnoticed, but the quality of the recovered image is high.

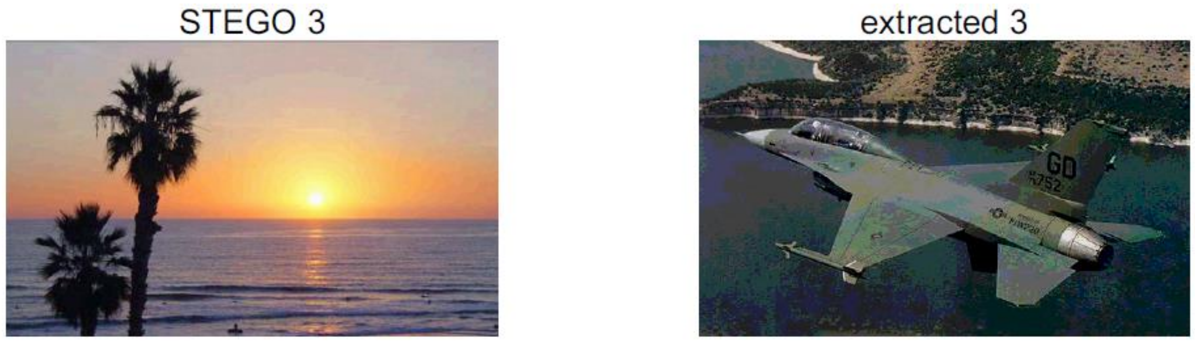

Fig. 5. Stego-image (STEGO 3) with hidden information and the extracted image (extracted 3) when using the three least significant bits

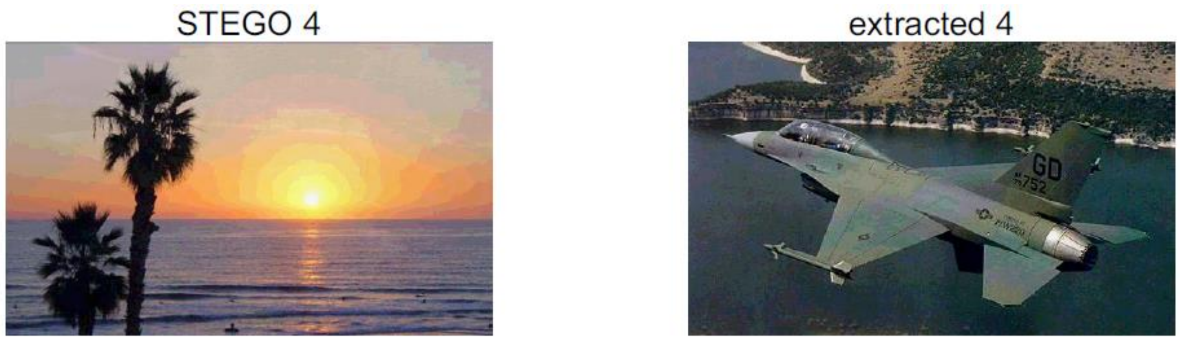

Fig. 6. Stego-image (STEGO 4) with hidden information and the extracted image (extracted 4) when using the four least significant bits

When using all eight bits of the COVER image to hide the hidden image, the resulting stego image is exactly the same as the hidden image.

4. Conclusion

The article investigates an algorithm for hiding one color image from BMP format to another color image, also in BMP format, using 24-bit color depth. The algorithm is programmatically implemented through MATLAB and Image Processing Toolbox. Extracting the hidden image loses some information, but there are a number of areas, such as aeronautics, meteorology and military missions, that require the hidden image to be restored 100%. This is achieved by applying the algorithm four times, using four "carrier" images, each of which carries two bits of the hidden image. In recovery, the extraction process is also applied four times.

Reference:

[1]. Antonov P., Malchev S. Kriptografiya v kompyutarnite komunikatsii. Varna, 2000, 315 p.

[2]. Paraskevov H., Zhelezov S., Tsankov Ts., Stanev S. Algorithm and program for steganography protection of text documents. Scientific Conference, dedicated to the 105th anniversary of the birth of the pioneers of computing John Atanasoff and John von Neumann, Shumen, 2008.

[3]. Terziev H., Savova Zh. Determining the amount of hidden information in jpeg images. International Scientific Conference “Defense Technologies” DefTech 2020, Faculty of Artillery, Air Defense and Communication and Information Systems, 2020, pp. 340-346, ISSN 2367-7902.

[4]. Terziev H., Stoyanova V. Steganalysis methods in images. International Conference on Technics, Technologies and Education ICTTE 2017, Yambol, Bulgaria, 2017, pp. 198-205, ISSN 1314-9474.

[5]. http://antivirus.trbk.net/bg/kompiutyrna-i-mrejova-sigurnost.html?start=31

[6]. http://www.garykessler.net/library/steganography.html