ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 1, Issue 9, December 2013

Message Encoding in Images Using Lifting Schemes

A.N.NAGA JYOTHI[1], Dr.G.A.E. SATISH[2]

M.Tech Student, Department of Electronics and Communications, Stanley Stephen College of Engineering and Technology, Kurnool, Andhra Pradesh, India[1]

Professor, Department of Electronics and Communications, Stanley Stephen College of Engineering and Technology, Kurnool, Andhra Pradesh, India[2]

Abstract: Digital images store large amounts of data and information. This data can be manipulated to some extend without being detected by human eyes. An example of such manipulations is insertion of secret information which is often referred to as information hiding. A successful insertion of a message into an image is more difficult using colour images than that of gray scale images. A successful information hiding should result in the extraction of the hidden data from the image with high degree of data integrity. This paper presents an information hiding technique that utilizes lifting schemes to effectively hide information in color images.

Keywords: PNG, Steganography, Haar lifting calculation

I. INTRODUCTION

The idea of sending secret messages is not new and in the digital age many methods of hiding messages are available [1]. Many different streams of digital media can be used as a cover stream for a secret message. Steganography is the art of writing secret messages so that only the sender and the intended recipient are aware of the hidden message. The hidden message might be written up, down, diagonally, or rotationally around a cylinder. The concept of steganography is also referred to as “hiding in plain sight” [2].

All of the steganography methods require a way to insert the message into the cover stream at the sending side, and a way to extract the message on the viewing side such that the message is not recognizable unless the hiding method is known. Some programs have been developed that analyse a media stream to determine whether any steganographic message has been added. Some of these steganalysis programs have reached high levels of accuracy in predicting which media files are “cover streams” and which ones are ordinary media files [3].

This paper introduces a method of secret message encoding that makes use of wavelets. Wavelets break down the stream into high and low frequency component parts called details and trends. As part of the research, an application called *Silent Words* was developed in Java that makes use of the integer-based wavelet transformation, lifting, and a Least Significant Bits (LSB) approach to hide messages in cover images. The lifting technique allows for variation in levels of transformation, selecting region of interest on cover image to be manipulated, type of wavelet transformation to be applied, and how far apart in the image each piece of the message is to be encoded. This method of encoding can be discovered if a bitwise comparison of the cover image is done with the steganographic image, which is the image that contains the secret message. Such a comparison may reveal manipulation but not the message. One way to thwart this discovery is to prevent access to the original image.

II. IMAGE FORMAT

Digital images are representations of two dimensional images using a binary format. For the purpose of this research, Portable Network Graphics (PNG) images are used that are of the true color image type. Each pixel in a true colour image has a section for the red channel, the green channel, and the blue channel. Additionally, pixels in the PNG format may specify a fourth channel called the alpha channel which stores the transparency of the pixel. Each channel contains the same number of bits (bit depth). A bit depth of 8 means that each channel of a pixel can contain a value in the range 0 to 255. The usual bit depths are 8 and 16 with 8 being the most common; however, other bit depths are possible. For this research, a bit depth of 8 is used for all images.

For true colour PNG images with a bit depth of 8, each pixel is stored using four bytes. There are three bytes that represent the colour channels and one byte that represents the alpha channel. As shown in Fig. 1, the alpha channel is stored in the leftmost byte in bit positions 24-31. The red channel is stored in the second byte in bit positions 16-23. The green channel is stored in the third byte in bit positions 8-15 and the blue channel is stored in the fourth byte in bit positions 0-7. Each of these channels has a possible value between 0 and 255 with 0 being black and 255 being pure red, pure green or pure blue depending on the channel in which the value resides. All images in this research are single layer images meaning that no composite image is used or created.

Channel	Alpha	Red	Green	Blue
Bit Position	31 24	23 16	15 8	7 0

Fig. 1. PNG Image Pixel Channels and Bits

A. Image Entropy

The definition of entropy is the measure of disorder or randomness in a closed system [4]. When applied to digital images, entropy is a measure of the amount of detail in the image. The application created from this research, computes an entropy value for the image selected by the user. The entropy value can be useful because it can provide an idea of how much detail is in the image before the user adds 'noise' to the image by inserting a message. Adding a message to an image modifies some of the bits of the pixels, thus modifying the randomness of the image as well. The entropy is calculated using 2.1.

$$Entropy = \sum_{j} \Pr(a_j) \log_2 \Pr(a_j), \qquad (2.1)$$

where a_j represents the j^{th} unique intensity value observed in a reading of the images pixel values and $\Pr(a_j)$ is the probability of that pixel value occurring within the image.

III. LIFTING SCHEMES

Lifting schemes, also known as integer-based wavelets, differ from wavelet transforms in that they can be calculated in-place. Similar to wavelet transformations, lifting schemes break a signal, the image, into its component parts 'trends' that approximates the original values and 'details' which refers to the noise or high frequency data in the image. A lifting scheme produces integers and this allows the original space to be used to hold the results [6]. Lifting operation requires two steps, one to calculate the trends and another to calculate the details. Fig. 2 illustrates that a pair of signal values is used to create a pair of coefficient values, the trends and the details.

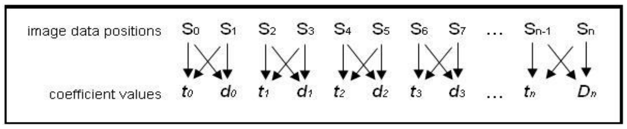
image data positions
S0 S1 S2 S3 S4 S5 S6 S7 … Sn-1 Sn
coefficient values
t0 d0 t1 d1 t2 d2 t3 d3 … tn Dn

Fig. 2. Coefficients Created from Signal Values

The trends and details, illustrated as *t* and *d* respectively, are produced in consecutive order after the transformation is applied to the signal as shown in Fig. 3.

coefficient values	*t0*	*d0*	*t1*	*d1*	..	..	*tn*	*dn*
image data positions	S0	S1	S2	S3	..	..	Sn-1	Sn

Fig. 3. Lifting Coefficients, One Level of Transformation

To separate the trends from the details as two component signals, a permutation step is done which results in array of coefficients that look like the one shown in Fig. 4.

coefficient values	*t0*	..	*tn*	*d0*	..	*dn*
image data positions	S0	..	S(n/2)-1	S(n/2)	..	Sn

Fig. 4. Lifting Coefficients after Permutation

When applied to a row of data, permutation moves all the trend pixels to the left, and the detail pixels to the right. This produces a meaningful view of the data. Because images are stored as two dimensional arrays, applying lifting schemes and permutations as described above must be done in two dimensions, once for rows and once for columns [7].

This research makes use of two lifting schemes: Haar, and Cohen-Daubechies-Feauveau CDF[2,2]. The Haar lifting calculation of the details and the trends for row *i* are shown in 3.1 and 3.2, respectively.

$$\text{detail: } s_{i,2j+1} = s_{i,2j+1} - s_{i,2j}, \; j = 0,1,..., n/2 \qquad (3.1)$$

$$\text{trend: } s_{i,2j} = s_{i,2j} + \frac{s_{i,2j+1}}{2}, \; j = 0,1,..., n/2 \qquad (3.2)$$

These calculations are done in the order shown above because the detail value calculated in 3.1 is used in the second term of 3.2.

The second lifting scheme is based on the CDF[2,2] wavelet. As shown in 3.3 and 3.4, the process for applying this lifting scheme on row *i* is slightly different from that of the Haar lifting scheme.

$$\text{detail: } s_{i,2j+1} = s_{i,2j+1} - \frac{1}{2}\left(s_{i,2j} + s_{i,2j+2}\right), \; j = 0,1,..., n/2 \qquad (3.3)$$

$$\text{trend: } s_{i,2j} = s_{i,2j} + \frac{1}{4}\left(s_{i,2j-1} + s_{i,2j+1}\right), \; j = 0,1,..., n/2 \qquad (3.4)$$

Note that the details from 3.3 are used in the calculation of trends in 3.4. This illustrates that once the transformation is complete, all the trends are located in one area, and the details are in the remaining area of the image matrix.

A. Level of Transformation

Lifting schemes can be applied on a signal more than once in successive applications of the transformation which are known as levels of transformation. In this research, a transformation applied to an image array at the first level would be applied to the entire array and a transformation applied at the second level would be applied to the trends section which is one-fourth of the original image. Therefore at the second level the outcome of the level one transformation is needed, and at level three the outcome of levels one two are required. This process is the same for higher levels of transformation. When described in this manner, it is apparent that lifting transformations are handled in successive order as illustrated in Fig. 5.

The goal is to hide data in the image in the lifting coefficients by manipulating the pixel values of these sections to hold the data and then reconstruct the image from these coefficients using inverse transformation. The Inverse transformation for the Haar lifting scheme is shown in 3.5 and 3.6.

inverse trend: $s_{i,2j} = s_{i,2j} - \frac{s_{i,2j+1}}{2}, j = 0,1,\ldots,\frac{n}{2}$ (3.5)

inverse detail: $s_{i,2j+1} = s_{i,2j+1} + s_{i,2j}, j = 0,1,\ldots,\frac{n}{2}$ (3.6)

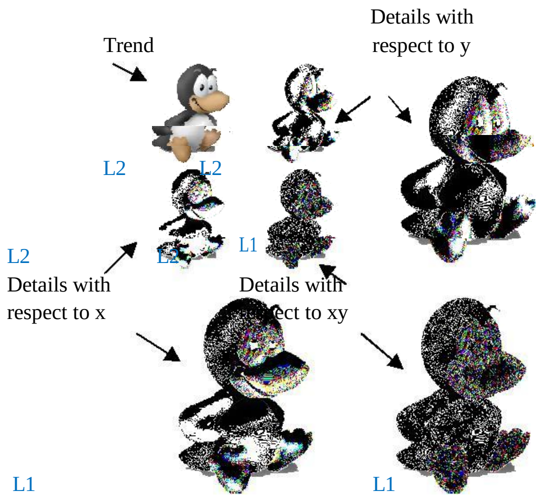
Trend
Details with respect to y
L2
L2
L2
L2
L1
Details with respect to x
Details with respect to xy
L1
L1

Fig. 5. Haar Lifting at Level 1 and Level 2 (L1 & L2)

The inverse CDF [2,2] lifting scheme is applied in 3.7 and 3.8. The inverse calculations for this scheme differ from the inverse Haar scheme calculations in that all the inverse trend calculations are performed first, and then all of the inverse details calculations are performed.

inverse trend: $s_{i,2j} = s_{i,2j} - \frac{1}{4}\left(s_{i,2j-1} + s_{i,2j+1}\right), j = 0,1,\ldots, n/2$ (3.7)

inverse detail: $s_{i,2j+1} = s_{i,2j+1} + \frac{1}{2}\left(s_{i,2j} + s_{i,2j+2}\right), j = 0,1,\ldots, n/2$ (3.8)

IV. INFORMATION HIDING PROCESS

Characters in the ASCII code can be represented using 8 bits. The value of lifting coefficients can be manipulated slightly without being noticed by visual inspection after the image is reconstructed using the manipulated lifting coefficients. This research project is base d on the premise that the bits of ASCII characters can be included in lifting coefficients without resulting in a visible appearance in the reconstructed image. Looking at a bitwise representation of an integer, the leftmost bit is called the most significant bit, or MSB. Conversely, the rightmost bit is known as the least significant bit or LSB. Because of this property, if the LSB were changed, the number would not be significantly affected. If the number was large, 1028 for example, and the LSB was changed from zero to one, the number would be changed from 1028 to 1029, which is a change of only 0.097%. Pixel values in an 8-bit gray-scale image usually range from zero to 255 inclusive for an 8 -bit image. If the LSB of 255 were changed to a zero from a one, the result would be 254, a change of 0.39%. As pixel values approach zero, the percent difference becomes larger.

The message hiding is done by changing pixel values, by no more than 3, to encode bits of the message. The next aspect of the method is to determine which specific bits to encode. Each pixel of a PNG image is made up of three channels, red, green and blue. The blue channel takes up bit positions 0 through 7, where 0 is the least significant or rightmost bit. The green channel takes up bit positions 8 through 15 and the red channel takes up bit positions 16
through 23. The lifting process is applied to all three of these channels individually. This is done to minimize distortion of the image that could be caused by overflow. While the PNG specification has three distinct colour channels in it, the
Channels are stored together in one integer type. *Silent Words* processes individual channels when inserting and extracting messages, but each pixel is stored as an integer data type in the program and in the image file.

The transformation category contains the options for which lifting transformations are available to the user, Haar or CDF [2,2]. Levels of transformation determine how much processing to be done on the image before the message is inserted. A Level 1 option results in applying the selected lifting scheme for one level, and then inserting the message. The level and technique is used to determine the maximum size of a message that can be en coded in the image. The maximum message size is calculated in 4.1 or 4.2.

$\max = (w * h)/(4^{n+1})$ (4.1)

$\max = 3(w * h)/(4^{n+1})$ (4.2)

where n is the transformation level and w and h represent the number of pixels in the width a nd height of the image, respectively. If the trends section of a distorted image is used, the maximum message size is calculated using 4.1. If the details section of a distorted image is used then the maximum size is calculated using 4.2.

A. Process of Adding a Message

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 1, Issue 9, December 2013

The process of adding a message to the pixels of an image is a multi -step process. In brief, an ASCII character stream is split into two-bit pairs, a lifting scheme is applied to an image, the two-bit pairs is inserted into the image in either the trends or details in the lifting domain, and then the inverse lifting process is applied to reconstruct the image. Figure 6 illustrates the process flow for encoding or decoding a message. As illustrated, the channels of the image pixels are split into separate arrays upon initialization. There is one array for each color channel and the alpha channel. Then it calculates the entropy and retrieves the text that has been entered by the user.

B. Encoding a Message in the Image Data

Before manipulating the array of characters that is read in, a terminator is added to the end of the array. The terminator is three '*' characters in a row. This terminator is used during the decode process to signal the end of the message input by the user in the encode process. The program then splits the ASCII character stream into 4 two - bit pairs per character. The two-bit pairs are created because the 2 LSBs of a pixel will be replaced with these two -bit pairs. Since all the characters in the English alphabet are contained within the lower 127 ASCII characters, only 4 two bit pairs are needed to represent each character. These two-bit pairs are stored in an array for later manipulation. The lifting scheme is then applied to the image down to the level specified by the user. Since the image is split into three colour channels, the lifting scheme must be applied three times, once for each color channel. The program automatically adjusts the encoding according to the lifting decomposition level.

Once the transformations are complete, the two-bit pairs of the ASCII characters are then hidden in the pixels of the processed image. The text offset value of the color channel specified by the user determines which bits are used to hide each subsequent two-bit pair of the ASCII character. Hiding the two-bit pair in the image is accomplished by overwriting the two selected bits of a pixel with the value of the two-bit pair. This is done by performing a bitwise AND operation with 0 and the two bits of the pixel, which effectively sets the two bits to 0. Then the two-bit pair to be hidden in this pixel is then combined with the pixel by a bitwise OR operator, effectively setting these pixel bits to the message bits.

Figure 7 illustrates an example of manipulating the two LSBs of a specified colour channel. In this example the green colour channel is being modified. One can see that if the original green pixel value had been 11111101, then the insertion of this two-bit pair of the message would not change the LSB values at all.

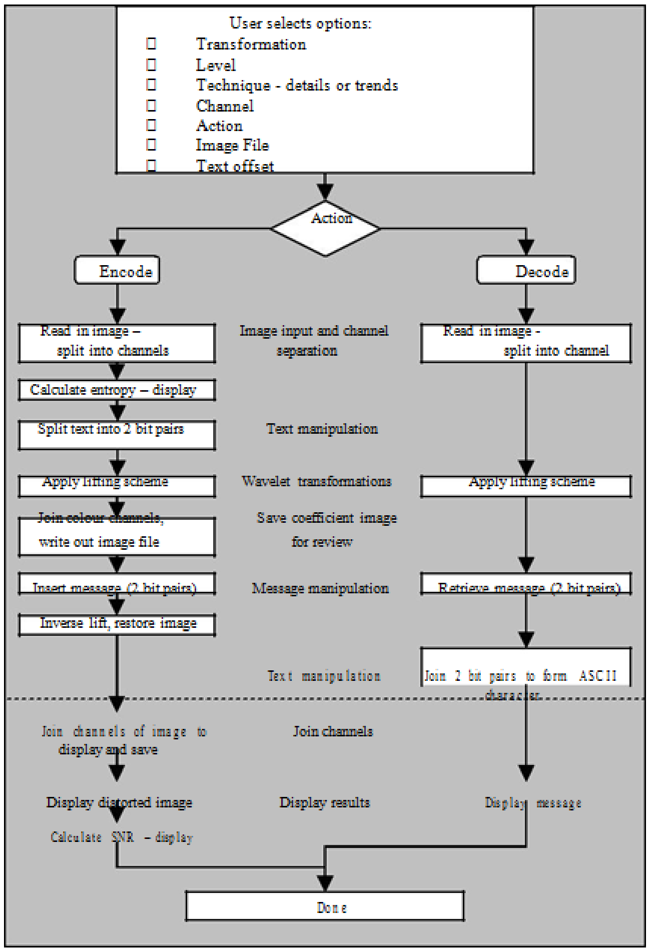
User selects options:
Transformation
Level
Technique - details or trends
Channel
Action
Image File
Text offset
Action
Encode
Decode
Read in image – split into channels
Image input and channel separation
Read in image - split into channel
Calculate entropy – display
Split text into 2 bit pairs
Text manipulation
Apply lifting scheme
Wavelet transformations
Apply lifting scheme
Join colour channels, write out image file
Save coefficient image for review
Insert message (2 bit pairs)
Message manipulation
Retrieve message (2 bit pairs)
Inverse lift, restore image
Text manipulation
Join 2 bit pairs to form ASCII character
Join channels of image to display and save
Join channels
Display distorted image
Display results
Display message
Calculate SNR – display
Done

Fig. 6. The Process Flow

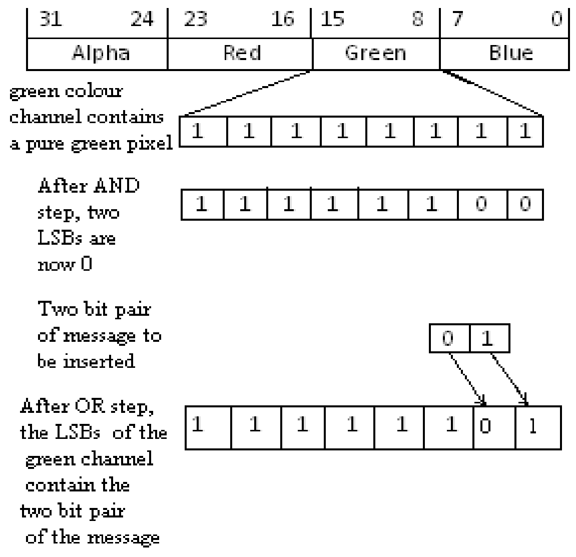
31 24 23 16 15 8 7 0
Alpha
Red
Green
Blue
green colour channel contains a pure green pixel
1 1 1 1 1 1 1 1
After AND step, two LSBs are now 0
1 1 1 1 1 1 0 0
Two bit pair of message to be inserted
0 1
After OR step, the LSBs of the green channel contain the two bit pair of the message
1 1 1 1 1 1 0 1

Fig. 7. LSB Bit Manipulation of the Green Colour Channel

Once these steps are complete, the inverse lifting scheme is applied to restore the image back to its original form.

V. DECODING A MESSAGE IN THE IMAGE DATA

As illustrated in Fig. 7, decoding a message that is inserted into an image requires fewer steps than to encode. The process flow starts out the same way as encoding with the user selecting the parameters that were used to encode the message. At this point the program splits the image into its color channels and applies the inverse lifting scheme to each channel to the level specified by the user. When the lifting transformation is completed, the program retrieves the message out of the pixels of the cover image.

VI. CONCLUSIONS

Images come in all variations and certain aspects of some images make them more suitable candidates than others for message encoding. The entropy of the image plays a major role in determining how suitable an image is for information hiding. The average entropy for the four images used was greater than 8. The two images with entropy values above that are both very good cover images for message encoding. One tested image with an entropy value below average encoded messages fairly well but only when using the trends section of the coefficients. As a general rule, with our approach, the higher the entropy in the signal image, the more suitable is the image for message encoding.

REFERENCES

[1] B. Weaver, Now You See It, Scientific Computing 24.6 (May 2007): 18- 39.
[2] B. Glass, Hide in Plain Sight, PC Magazine 21.18 (15 Oct. 2002): 75.
[3] Tucker, Patrick. "Hiding Secrets in Computer Files." Futurist 40.5 (Sep. 2006): 12-12.
[4] R. Gonzales, and R. Woods, Digital Image Processing, Addison Wesley Publishing Co., 1993.
[5] C. Birslawn, Fingerprint Go Digital, Notices of American Mathematical Society, Vol. 42, No.11, P. 1278-1283, Nov. 1995.
[6] W. Sweldens, Building Your Own Wavelets at Home, Wavelets in Computer Graphics, ACM SIGRAPH Course Notes, 1996.
[7]A. Calderbank, I. Daubechies, W. Sweldens, and B. Yeo, Wavelet Transforms that Map Integers to Integers, Mathematics Subject Classification, 42C15, 94A29,1996.