Robust Image Steganography In Transform Domain

**Abstract**

In this paper, we describe an image steganography technique using a modified version of the haar wavelet transform and compare its effectiveness with other steganography techniques.

**Keywords**

Steganography, DWT, Haar Wavelet Transform, LSB

**1. Introduction**

Steganography means "covered writing" in Greek. The main purpose of digital steganography is to convey information in secret by embedding it in other media such as images, such that there is no perceptible difference in the resulting image.

In this paper, we will use the term *secret image* to refer to the image to be hidden, *cover image* to refer to the image in which we will store the secret in, and *stego image* to refer to the resultant image when the secret image is stored in the cover image.**2. Background**

**2.1 Least Significant Bit**

Many image formats have precision levels that are far from perceivable by the average human vision[1]. This means that small variations in the colors of an image are imperceptible to the human eye. Conventional LSB techniques replace the last bit of each pixel in the cover image.

While LSB steganography results in a stego image that is near imperceptible from the cover image, it is not robust to attacks such as JPEG compression.

**2.2 Discrete Wavelet Transform**

The Haar transform is one of the simplest DWT.

**[More explanation of haar]**

**2.2.1 1D Haar Transform**

In 1D Haar transform, a signal s is decomposed into two sets of coefficients, the approximation and detail coefficients, denoted by E and F respectively in Figure 1.

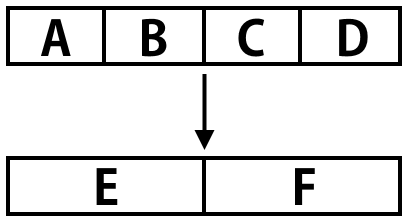


Figure 1: 1D Haar Wavelet Transform

These coefficients are calculated as follows:

Ei = ( S2i + S2i+1 ) / 2

Fi = ( S2i - S2i+1 ) / 2

Where S is the signal, E and F are the approximation and detail coefficients respectively, and 0 < i < length of S

**2.2.2 2D Haar Transform**

In the context of images, the 1D Haar transform is generalized to 2D. The 2D Haar transform decomposes an image into 4 sub-bands [2]:

1. A (Approximation Area) – holds global properties of the image
2. H (Horizontal Area)– holds horizontal details
3. V (Vertical Area) – holds vertical details
4. D (Diagonal Area) – holds diagonal details

To compute the 2D Haar transform of an image, take the 1D Haar transform along the rows, then take the 1D Haar transform along the columns. For an M x N image I, this gives the following formulae:

Ai, j = (I2i, 2j + I2i, 2j+1 + I2i+1, 2j + I2i+1, 2j+1 ) / 4

Hi, j = (I2i, 2j - I2i, 2j+1 + I2i+1, 2j - I2i+1, 2j+1 ) / 4

Vi, j = (I2i, 2j + I2i, 2j+1 - I2i+1, 2j - I2i+1, 2j+1 ) / 4

Di, j = (I2i, 2j - I2i, 2j+1 - I2i+1, 2j + I2i+1, 2j+1 ) / 4

**3. Proposed Method**

One of the biggest disadvantages of the Haar transform is that the coefficients from the decomposition are not integer values. In embedding the secret data in the coefficients and

performing the inverse Haar transform, there is some loss of precision. In this section, we propose a simple modification to the Haar transform so that it provides a mapping from integer to integer.

**3.1 Modified Transform**

In the original Haar Transform, for each block of 2 x 2 pixels, the coefficients are calculated by taking the average of the sum and/or differences of the pixel values. In our modified version, we simply remove the averaging function. This gives the formulae:

Ai, j = I2i, 2j + I2i, 2j+1 + I2i+1, 2j + I2i+1, 2j+1

Hi, j = I2i, 2j - I2i, 2j+1 + I2i+1, 2j - I2i+1, 2j+1

Vi, j = I2i, 2j + I2i, 2j+1 - I2i+1, 2j - I2i+1, 2j+1

Di, j = I2i, 2j - I2i, 2j+1 - I2i+1, 2j + I2i+1, 2j+1

The inverse transform is calculated as follows:

I2i,2j = (Ai,j + Hi,j + Vi,j + Di,j ) / 4

I2i+1,2j+1 = (Ai,j - Hi,j - Vi,j + Di,j ) / 4

I2i+1,2j = (Ai,j - Vi,j + 2I2i+1,2j+1 ) / 2

I2i,2j+1 = (Ai,j - Hi,j + 2I2i+1,2j+1 ) / 2

****

Figure 2: (a) original image (b) result after modified transform

**3.2 Embedding Procedure**

Step 1 : Split both the cover and secret images into individual RGB channels.

Step 2: Perform the modified transform on the blue channel of the cover image.

Step 3: For a pixel in the blue channel of the secret image, we segment the 8-bit pixel value into 4 blocks of 2 bits – A, B, C and D, where A is the most significant bits, and D the least.

Step 4: Replace the two least significant bits of one of the approximation coefficient with A. Repeat this step for the horizontal, vertical and diagonal areas, replacing the bits at the same index with B, C and D respectively. The most significant bits of a pixel in the secret image are stored in the approximation coefficients, and subsequent bits are stored in the horizontal, vertical and diagonal coefficients. This takes advantage of the fact that the approximation coefficient is more robust to changes. As such, hiding the most significant bits in the approximation coefficients will result in less noise.

Step 5: Repeat Step 3 to 4 until all of the bits in the secret image has been stored.

Step 6: Apply the inverse function on the modified coefficients to obtain the blue channel of the stego image.

Step 7: Repeat Step 2 to 6 with the red and green channels. Merge the 3 resulting channels to obtain the final stego image.

For each pixel in the transformed cover image, 2 bits of the secret image are stored. As such, the proposed method is able to store a secret of up to a quarter the size of the cover image.

As can be seen from Figure 3, the stego image is near imperceptible from the cover image.

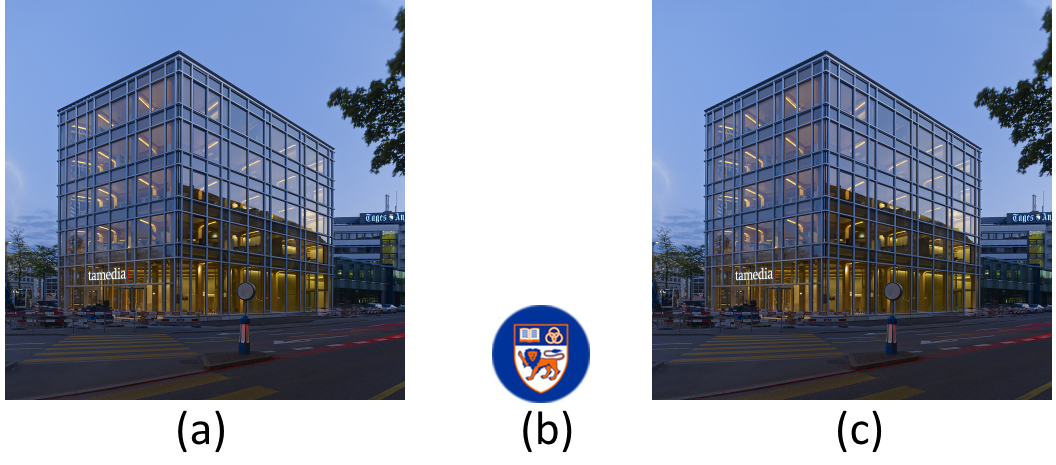


Figure 3: (a) cover (b) secret (c) stego image

**3.3 Extraction Procedure**

Step 1: Split the stego image into three individual RGB channels.

Step 2: Perform the modified transform on the blue channel.

Step 3: Extract the third and fourth least significant bits from each coefficients of the approximation, horizontal, vertical and diagonal area at the same index. Combine the extracted bits to obtain the pixel value. Repeat until the whole image is obtained.

Step 4: Repeat Steps 2 to 3 for the red and green channels. Combine the results to obtain the extracted secret.

Figure 4 shows the difference between the secret image and the result of the extracted image. Since only RGB values are stored in the cover image, any transparency property that the secret image holds is lost.



Figure 4: (a) secret image (b) extracted secret

**3.4 A More Robust Approach**

To increase the robustness towards attacks such as JPEG compression, the embedding process is done on the second level decomposition instead of the first level.

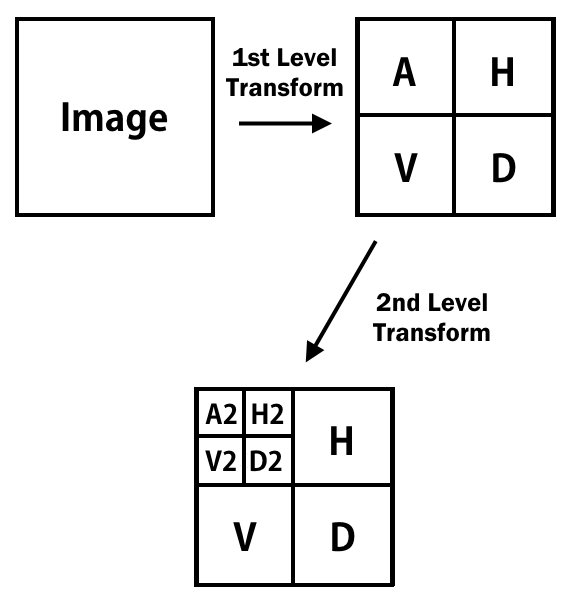
****

Figure 5: 2nd level 2D transform

To obtain the second level transform, the same transform procedure is performed on the approximation area of the first level decomposition as shown in Figure 5. The embedding process is then done on the coefficients of the second level decomposition. This reduces the embedding capacity by half, but increases the robustness.

After embedding the secret in the coefficients of the second level decomposition, the inverse transform does not necessarily produce an integer. As such, there may be truncation of floating point values when saving the image. This results in more noise in the extracted secret image. However, this process is more robust than the first level embedding method in that features of the image are still visible after certain levels of attacks such as compression.

REFERENCES

1. Champakamala .B.S, et al., *Least Significant Bit algorithm for image steganography*, International Journal of Advanced Computer Technology
2. Piotr P. and Agnieszka L., *The Haar–Wavelet Transform in Digital Image Processing: Its Status and Achievements*, Univeristy of Silesia