AN EFFICIENT HASH BASED STEGANOGRAPHY TECHNIQUE FOR TEXT MESSAGE USING COLOR IMAGES

Saurabh Singh¹, Dr. Ashutosh Datar²

¹PG Student, Department of Electronics & Communication Engineering, S.A.T.I. Vidisha, INDIA (M.P.)
²Head, Department of Bio-Medical Engineering, S.A.T.I. Vidisha, INDIA (M.P.)

ABSTRACT

In this paper, a hash based approach for color image Steganography using canny edge detection method is proposed. One of the advantages of using edge detection technique is to secure the data. For encoding the text data in an image Steganography procedure is followed. Edge detection is done by canny method and then hash function is used to embed text data in the RGB color image. The hash is a fast and secure approach for image Steganography. This found that canny edge detection offers superior performance for detecting edges in an image. Large edge detected image is preferred for secure Steganography. The proposed method provides a better security and supports different types of file format like-jpg, jpeg, bmp, tiff etc.

Keywords: Edge detection, Steganography, Encoding & Decoding, Hash function.

I. INTRODUCTION

Steganography (Steganos-“Covered”, Graphie-“Writing”) is the art and science of writing hidden messages in such a way that no one apart from the intended recipient knows of the existence of the message [1]-[5]. The first step in embedding and hiding information is pass both the secret message and the cover message into the encoder. Inside the encoder, one or several protocols may be implemented to embed the secret information into the cover message. The type of protocol will depend on the information in fig.1. contents to be embedded.
For example, an image protocol is used to embed information inside images. A key is often needed in the embedding process. This can be in the form of public or private key to encode the secret message with sender private key and the recipient can decode it using sender public key. In embedding the information this way, the chance of a third party attacker getting hold of the Stego-object and decoding it to find out the secret information is greatly reduced.

Hence, the embedding process inserts a mark \( M \) in the object \( I \). Random number generator is generally used to produce a key \( K \), which is embedded in \( I \). Hence, the marked object \( I \) is generated as:

\[
I \times K \times M - I
\]

The output of the encoder will be a Stego object contains original cover image and embedded secret information. The Steganography algorithm must ensure that the Stego object bear a very close resemblance to original cover object. The Stego object then may be sent through usual communication channels to intended recipients. The Stego object at receiver end must be decoded to retrieve the secret information. In decoding process, Stego object is decoded with the help of the public or private key. Depending upon the encoding technique, sometimes the original cover image is also needed in decoding process. After successful decoding, the embedding secret information may be extracted and viewed. There are many algorithms available for image Steganography. eg. LSB (Least significant bit) method, masking etc. LSB method is simple and popular method but its security aspect is poor. Hash function based approach is robust and convenient. A hash function converts variable lengths data of fixed length. The hash function returns hash codes or checksum or simple hashes. [Some more about hash function coding and decoding].

II. **EDGE DETECTION**

Edge in an image contains shape information. Edges represent abrupt transition of gray level in an image. The edge detection is the process of finding such transition, which represents some physical boundary in the scene [6]–[11]. The goal of edge detection is to obtain a line drawing of the scene in an image, so as to further extract information regarding important features [Fig.2.]. This information may then be used in pattern recognition algorithms. Below figure shows the edge detection example-
Figure 2. Edge detection example

Basic steps in edge detection:
1) Smoothing: Noise removal/suppression with preservation of edges.
2) Enhancement: Sharpening of edges by filtering.
3) Detection: Determination with certainty if pixel belongs to an edge or not.
4) Localization: Determining exact location of edge.

There are various operators which are used in edge detections. Some noteworthy edge detectors are:

1) Robert Edge Detection
   It is one of the early detectors, which is simple and efficient. It highlights regions of high spatial frequency which may usually correspond to edges. It then assigns pixels values in output images; correspond to estimated complete magnitude of spatial gradient of the input image.

\[
G_x = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}
\]

2) Sobel Edge Detection
   It is central difference based operator giving higher weight to central pixel. It is an approximation of first order derivative of Gaussian kernel. Sobel operators which are shown in the masks below:

\[
G_x = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}
\]

3) Prewitt Edge Detection
   It is also based on central difference where operator measures two components. Vertical edge components and horizontal edge components for current pixel.

\[
G_x = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}
\]
4) Kirsch Edge Detection

This operator or compass kernel is a non-linear edge detector. It finds maximum edge strength in predetermined direction. It is represented by the mask:

\[
E = \begin{bmatrix}
-3 & -3 & 5 \\
-3 & 0 & 5 \\
-3 & -3 & 5
\end{bmatrix}
\]

\[
NE = \begin{bmatrix}
-3 & 5 & 5 \\
-3 & 0 & 5 \\
-3 & -3 & -3
\end{bmatrix}
\]

\[
N = \begin{bmatrix}
5 & 5 & 5 \\
-3 & 0 & 5 \\
-3 & -3 & -3
\end{bmatrix}
\]

\[
NW = \begin{bmatrix}
5 & 5 & -3 \\
5 & 0 & -3 \\
-3 & -3 & -3
\end{bmatrix}
\]

\[
W = \begin{bmatrix}
5 & -3 & -3 \\
5 & 0 & -3 \\
5 & -3 & -3
\end{bmatrix}
\]

\[
SW = \begin{bmatrix}
-3 & -3 & -3 \\
5 & 0 & -3 \\
5 & 5 & -3
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
-3 & -3 & -3 \\
-3 & 0 & -3 \\
5 & 5 & 5
\end{bmatrix}
\]

\[
SE = \begin{bmatrix}
-3 & -3 & 5 \\
-3 & 0 & 5 \\
-3 & 5 & 5
\end{bmatrix}
\]

5) LoG (Laplacian of Gaussian) Edge Detection

This operator has smoothing effects through convolution with Gaussian shape kernel. It is followed by application of laplacian operator.

\[
G_x = \begin{bmatrix}
0 & -1 & 0 \\
-1 & 4 & -1 \\
0 & -1 & 0
\end{bmatrix}
\]

\[
G_y = \begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1
\end{bmatrix}
\]

6) Canny Edge Detection

Canny edge detector is an optional edge detector algorithm which has good detection, good localization capability with minimal response. The problem with above edge detector is that the detected edge may not be complete due to noise, breaks in edges from non-uniform illumination, unwanted effects due to spurious intensity is discontinuity etc. Hough transform is one of the popular approaches to link edge pixels into meaningful edges. The Hough transform [12]-[15] has two steps:

1. Peak detection- For each detected peak the location of non-zero pixels that contributes to that peak is determined.
2. Line detection and linking: After the peaks are identified, there corresponding line segments with starting and ending points are determined.

Figure 3. Hough transform based edge detection
III. HASH FUNCTION

To create a digest of the message, hash function [16]-[18] is used. The hash function creates a fixed digest from a variable–length message (fig.4).

![Figure 4. Signing the digest](image)

The two most common hash functions are called MD5 (Message Digest 5) and SHA-1 (Secure Hash Algorithm 1). The first one produces a 120-bit digest. The second produces a 160-bit digest. Hash functions have must two properties to guarantee it success.
1. Hashing is one-way; the digest can only be created from the message, but not vice-versa.
2. Hashing is one-to-one function; there is little probability that two message will create the same digest.

One practical use is a data structure called a hash table where the data is stored associatively. Searching for a person's name in a list is slow, but the hashed value can be used to store a reference to the original data and retrieve constant time (barring collisions). It is easy to generate hash values from input data and easy to verify that the data matches the hash, but hard to 'fake' a hash value to hide malicious data. Hash functions are also used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicated or similar records in a large file, finding similar stretches in DNA sequences, and so on.

Different types of hash function are available but most types of hashing function the choice of the function depends strongly on the nature of the input data. Types of hash function-

1. Trivial hash function
2. Perfect hash function
3. Minimal perfect hash function
4. Cryptographic hash function
5. Hashing with checksum
6. Hashing variable length data
7. Modulus hash function etc.

In this paper, Modulus hash function has been used that is the mod of the division hash function could be \( h = z \mod n \) (the remainder of \( z \) divided by \( n \)). This is the combination of addition, subtraction, multiplication and division function.

It is of two types:
1. Symmetric hash function-This type of hash function ensure that sender and receiver used the same key for data encoding and decoding.
2. Asymmetric hash function- This type of hash function ensure that sender and receiver used the different keys for data encoding and decoding.

In our methodology, symmetric modulus hash function has been used for text data hiding and retrieval purpose in the Red, Green and Blue pixels of the image.
IV. APPROACH FOR STEGANOGRAPHY

This approach allows the user to embed their secret textual information in images in a way that can be invisible and doesn't degrade or affect the quality of the original image [2]-[4]. Users want to make their information secure or protect their work from piracy. This approach is able to manipulate with different file formats e.g. bitmap, jpeg, jpg, gif and tiff etc.

The Steganography steps for the RGB image used in this work are (fig.5).

1. Input Image-An input interface is provided so that a user can input a (bmp, jpg, gif or tiff etc.) image for hiding personal data for privacy purposes.
2. Input Textual Data-Input the text file containing the textual data which the user wants to code in the image. The input text file is read by the system.
3. Coding Data in Image-For coding textual data in the image, a hash-based algorithm is used. Basic purpose of using the hash-based algorithm is to pick pixels randomly to store text data. The text data are stored in red, blue and green pixel of the employed RGB image. Once the image is encoded, it is transmitted to intended user.
4. Decoding Data from Image-For decoding textual data in the image, the hash key is used that was generated during coding. The pixels (red, green, blue byte) values of each position are read one by one and generated characters concatenated to form a complete message string.

V. PROPOSED METHODOLOGY

The image and the message are encoded at the sender end. The flow chart for encodings and decoding are given in fig (6) & fig (7) respectively [19]–[25].

1. Encoding Procedure:
   The encoding procedure involves of the RGB image and message string. Application of canny operator provides edge information, which is linked by using Hough transform. The edge pixels are identified and are used as a hash key (K). The hash function uses hash key K, input image and text data to generate pattern. This pattern contains the information regarding position of pixels for hiding message data. The pixels are chosen from Red, Green and Blue plane of the image. The output contains encoded image (Stego-object) which is sent to the intended user. The flow chart of the encoding procedure shown in fig. (6).
Figure 6. Flow chart of the encoding procedure

1. Start
2. Read RGB image
3. Edge identification and linking by Canny edge detection & Hough transform
4. Pixel identification, Hash key
5. Read message string L
6. Hash function & hash digest, I
7. Check remainder value, in Mod function
   - If I ≤ L
     - Modify green pixel
     - Modify blue pixel
     - Modify red pixel
     - Generate Stego-object
     - End
   - Else
     - Modify green pixel
     - Modify blue pixel
     - Modify red pixel
     - Generate Stego-object
     - End
2. Decoding Procedure: It is the reverse process of encoding. In this process received Stego-object contains original image along with hidden message, which is to be extracted. The input hash key is used to generate hash values, which contain information about location of pixels where message is stored. The message string is extracted from Red, Green and Blue pixels of the image. The output contains extracted text message. The flow chart of decoding the Stego-object is shown in fig. (7).

![Flow chart of decoding procedure]

**Figure.7.** Flow chart of the decoding procedure.
VI. IMAGE QUALITY PARAMETER

Image quality is a characteristic of an image that measures the perceived image degradation (typically, compared to an ideal or perfect image) [5]. Two parameters are there:

1. **MSE (Mean square error)**
   
   It is defined as the squared difference between the original image and estimated image.
   
   $$MSE = \frac{1}{N} \sum_{i=1}^{N} [X - \hat{X}]^2$$

   Where: $X =$ original value, $\hat{X} =$ stego value and $N =$ number of pixels

2. **PSNR (Peak signal to noise ratio):**

   Peak Signal-to-Noise Ratio, often-abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation [10]. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. PSNR is most easily defined via the mean squared error (MSE):

   $$PSNR = 10 \cdot \log_{10}\left(\frac{L^2}{MSE}\right)$$

   $$= 20 \cdot \log_{10}\left(L^2 / \sqrt{MSE}\right)$$

   Where: $L =$ maximum value, $MSE =$ Mean Square Error

VII. SIMULATION RESULTS

In the proposed methodology three different color image Lena, Peppers and Baboon of standard size are used. Simulation results are performed in Matlab2010a version. A comparative of test images under simulation and encoded images corresponding histograms, detected edges are shown in fig (8) - fig (12).

The quantitative results are shown in Table-I. On making comparison between original images and encoded images and their respective histograms. It is evident that no noticeable change is observed. Hence, it is also observed that the Steganography process did not significantly altered the image qualities. It is also observed from fig. (10) that Baboon.jpeg offers maximum edge pixels while the Peppers.jpeg has the lower edge pixels. Hence the PSNR of Baboon.jpeg is higher that two others except for first case when Lena.jpeg has high value in fig. (13). It may be due to small color variation in Lena.jpeg. Further, a comparison of results of proposed approach is made with Kritika et.al. [19]. The proposed method shown superior results in all cases under simulation.

![Figure 8. Original images of size 512x512](image_url)
Figure 9. Encoded Images with Text Data (2547 Bytes)

Figure 10. After Applying Canny Edge Detection Algorithm

Figure 11. Histogram of original image

Figure 12. Histogram of encoded image with text data (2547 Bytes)
Table I
PSNR Output for text encoding

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Message length</th>
<th>Lena (512 × 512)</th>
<th>Peppers (512 × 512)</th>
<th>Baboon (512 × 512)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Kritika et.al. [19]</td>
<td>Proposed</td>
<td>Kritika et.al. [19]</td>
</tr>
<tr>
<td>1.</td>
<td>849 Bytes</td>
<td>46.7704</td>
<td><strong>47.5599</strong></td>
<td>42.4704</td>
</tr>
<tr>
<td>2.</td>
<td>1698 Bytes</td>
<td>43.1161</td>
<td><strong>43.7728</strong></td>
<td>39.8468</td>
</tr>
<tr>
<td>3.</td>
<td>2547 Bytes</td>
<td>40.4854</td>
<td><strong>41.5473</strong></td>
<td>37.9358</td>
</tr>
<tr>
<td>4.</td>
<td>3396 Bytes</td>
<td>39.5810</td>
<td><strong>40.0503</strong></td>
<td>36.7382</td>
</tr>
<tr>
<td>5.</td>
<td>4287 Bytes</td>
<td>38.5342</td>
<td><strong>39.6310</strong></td>
<td>35.7352</td>
</tr>
</tbody>
</table>

Graphical Results of Calculated PSNR

Figure 13. The graph shows that as the message length increases PSNR decreases, baboon image gives better results than Lena and peppers image
VIII. CONCLUSION

The work proposed an approach for Steganography using RGB image and text message. The results, in comparison to [19] are superior. In this work, all RGB color planes of image are used for hiding secret text message without loss of image quality. Hence, this approach offers high message carrying capacity and high-encoded image quality. The Human Visual System (HSV) cannot detect these fine changes in original image, made due to Steganography process. Moreover, use of hash function improves its robustness. Hence, it may be adopted for Steganography using color images. This Color image (RGB) Steganography technique can be extending to further audio, video etc.

ACKNOWLEDGMENT

The author wish to acknowledge Dr. S.N. SHARMA (Head, Department of E & C Engineering) for his support.

REFERENCES


