Image Steganography Using an Edge Detection-Driven LSB Technique

1st Abhishek Bhamare
Information Technology
National Institute of Technology Karnataka
Surathkal, India

2nd Sankethraj Kotagond
Information Technology
National Institute of Technology Karnataka
Surathkal, India

3rd Prajwal Gabhane
Information Technology
National Institute of Technology Karnataka
Surathkal, India

4th Rajesh Kumar
Information Technology
National Institute of Technology Karnataka
Surathkal, India

***Abstract*—This project explores an approach to image steganography focused on minimizing loss in the quality of the cover image by embedding secret content (either text or an image) within its edges. Traditional steganographic methods often alter the visible characteristics of the cover image, compromising its integrity. Our method addresses this by first detecting the edges within the cover image using edge detection techniques and then embedding the hidden content along these edges. This strategic embedding reduces visual distortion, ensuring the hidden data remains imperceptible while maintaining high quality in the cover image.**

To evaluate the effectiveness and resilience of our approach, we employed Histogram Analysis, Chi-square test analysis, Structural Similarity Index(SSIM), Mean Squared Error (MSE) and Peak Signal to Noise Ratio(PSNR) to compare the original and stego images for both embedding scenarios (text in image and image in image). These analyses revealed that while visual quality was preserved, certain statistical differences were detectable, exposing vulnerabilities in the steganographic method.

***Index Terms*—Image steganography, Edge Detection based LSB**

I. Introduction

In the era of rapid digital communication, the internet has revolutionized the way information is shared. While this has brought immense benefits, it also presents significant challenges in securing sensitive data over open networks. The security of digital messages is critical to prevent unauthorized access, and several techniques have been developed to protect information from malicious entities. Cryptography is one such method, which ensures the confidentiality of data by allowing only the sender and the intended recipient to view its content. However, along with cryptography, information-hiding techniques like steganography and watermarking also play vital roles in securing digital data.

Watermarking and steganography both obscure confidential information within seemingly innocuous media, such as images, videos, audio, and text. However, their purposes and applications differ. Watermarking is primarily used for authentication and copyright protection of digital content, ensuring the ownership of the media. In contrast, steganography is the art of concealing information in such a way that it remains undetected, making it particularly useful for hiding sensitive data. In steganography, the ability to embed information without altering the perceptibility of the host medium is of utmost importance.

Traditional visible watermarking often damages the structure of the image, making the embedded information visible and vulnerable to removal. On the other hand, steganography allows for embedding information within digital media in a manner that is imperceptible to the human eye, offering a more secure alternative. This capability has made steganography increasingly popular in fields like copyright protection, secure data transmission, and covert communication.

In this project, we design an image steganography technique based on an edge detection-based LSB algorithm that minimizes the loss of quality in the cover image while embedding secret content, which can be either text or an image. The approach focuses on embedding the hidden data along the edges of the cover image to reduce visual distortion. By first detecting the edges using edge detection techniques, the method ensures that the secret content is seamlessly embedded along these boundaries, maintaining the hidden information's imperceptibility and preserving the cover image's high visual quality.

To assess the fidelity of the stego image and detect any artifacts introduced by embedding, we utilized histogram analysis, chi-square test analysis, Peak Signal to Noise Ratio, Structural Similarity Index Measure and Mean Squared Error.

Histogram analysis:
It evaluates the distribution of pixel intensities in the cover and stego images. The embedding process can alter the distribution subtly, and these changes, although imperceptible

to the naked eye, may indicate the presence of hidden data. The histograms of the stego images demonstrated minor deviations in pixel intensity distributions, particularly in regions with dense embedding. These deviations suggest the presence of detectable artifacts, albeit subtle.

Chi-Square Test Analysis:
The chi-square test is a statistical tool that quantifies the deviation between the expected and observed frequency distributions of pixel values. For the chi-square test, the test statistic is computed as:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

For the observed frequency, use O_i. For the expected frequency, use E_i. The chi-square statistic is denoted as χ^2.

Structural Similarity Index (SSIM):
SSIM evaluates image quality by measuring the structural similarity between two images. It considers luminance, contrast, and structural information. SSIM values range from 0 to 1, where 1 indicates identical images. It is particularly useful for assessing perceived image quality and is more aligned with human visual perception. It provides a more holistic comparison by considering the structural details and is better for capturing visual similarity.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Where: μ_x and μ_y: Mean intensities of images x and y. σ_x^2 and σ_y^2: Variances of x and y. σ_{xy}: Covariance of x and y. C_1 and C_2: Small constants to stabilize the division.

Mean Squared Error (MSE):
MSE calculates the average squared difference between corresponding pixels in two images. A lower MSE value indicates greater similarity. However, it doesn't account for perceptual quality, as it treats all pixel errors equally regardless of their context within the image. It offers a straightforward numerical difference measure but is less useful for perceptual analysis.

$$\text{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left(I(i,j) - K(i,j)\right)^2$$

Where $I(i, j)$ is the pixel value of the original image, $K(i, j)$ is the pixel value of the processed image, and M, N are the dimensions of the image.

Peak Signal to Noise Ratio (PSNR):
PSNR measures the quality of a reconstructed or compressed image compared to the original. It quantifies the ratio between the maximum possible pixel value of an image and the power of the noise affecting its quality. PSNR is measured in decibels (dB). A higher PSNR value indicates better image quality. It is widely used in image processing as a benchmark for comparing compression or embedding techniques. It is effective for objective quality comparison but can be less accurate in predicting perceptual quality.

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

Where MAX is the maximum possible pixel value of the image (e.g., 255 for 8-bit images).

Our vulnerability analysis revealed that while the proposed edge-based embedding scheme maintains high visual fidelity, statistical discrepancies introduced during the embedding process can be exploited by advanced steganalysis techniques.

II. Literature Survey

Image steganography in the spatial domain involves embedding data directly into pixel values, with techniques such as Least Significant Bit (LSB) substitution, adaptive steganography, and other pixel-based approaches being widely used. These methods offer advantages like simplicity and high embedding capacity, but they are prone to statistical attacks and lack robustness against image processing operations. Applications of spatial domain steganography include secure communication, watermarking, and copyright protection. Future advancements are needed to enhance security, capacity, and imperceptibility while addressing vulnerabilities. [1]

An image steganography method is presented that combines Canny edge detection, dilation operators, and hybrid coding techniques. The Canny edge detection algorithm identifies significant edges in an image, providing optimal locations for data embedding. The dilation operator is applied to enhance the robustness of the steganographic method, making it less susceptible to image distortions. Hybrid coding techniques are used to optimize the data hiding capacity while reducing the visual impact on the host image. The proposed approach improves both the security and efficiency of steganography for hidden communication in digital images. [2]

A novel method for steganography based on Least Significant Bit (LSB) matching is presented, focusing on improving the robustness and security of traditional LSB-based techniques. The approach revisits existing methods and introduces enhancements to reduce detectability while maintaining a high capacity for data embedding. The improvements aim to address the vulnerabilities in conventional steganographic algorithms, ensuring better image quality and more secure information concealment. Experimental results validate the effectiveness of the proposed method, highlighting its potential for secure communication and data hiding applications. [3]

An image steganography method using deep learning-based edge detection is introduced, focusing on embedding data securely within the edges of an image. The approach utilizes deep learning models to identify key features and edges in

an image more accurately than traditional methods, enhancing the precision of data hiding. By embedding information along the detected edges, the method ensures minimal distortion of the host image, making the hidden data more secure and less detectable. The technique is evaluated for its effectiveness in maintaining image quality while maximizing the capacity for data embedding, making it a strong solution for secure image-based communication. [4]

An improved LSB (Least Significant Bit) algorithm for image steganography is proposed, focusing on enhancing accuracy and reducing distortion in the embedding process. The method addresses challenges such as measurement uncertainty and the potential for detectability in traditional LSB techniques. By using an optimization model combined with a random search algorithm, the approach refines the steganographic process to improve embedding capacity while minimizing image distortion. The method is also applicable to fields like copyright protection, where maintaining the integrity of the host image is crucial. The results show significant improvements in both data embedding accuracy and image quality. [5]

III. Methodology

This section outlines the steps and techniques used for embedding and extracting secret messages and images using an edge-detection based LSB approach. The proposed method employs edge detection to selectively embed data into edge regions, ensuring imperceptibility while maintaining the visual quality of the cover image. The below proposed method is figuratively shown in Fig 1

Embedding Process :

1) Input: Cover image, hidden image/text.
2) Steps:
   - Convert images to RGB and NumPy arrays.
   - Flatten hidden image pixels into binary strings. If the data to be hidden is text, then simply convert it into a binary string.
   - Process the cover image in blocks (*block_width* $\times$ *block_height*):
     - Detect edges using gradients (G_x, G_y).
     - Embed more bits in edge regions and fewer bits in non-edge regions.
   - Modify the LSBs of R, G, and B channels using `embed_bits()`.
3) Output: Stego-image with hidden data.

Extraction Process :

1) Input: Stego image, hidden image dimensions/text length.
2) Steps:
   - Process blocks in encrypted image.
   - Detect edges and extract LSBs using `extract_bits()`.
   - Combine bits to reconstruct the hidden image/hidden text.
3) Output: Reconstructed hidden image/hidden text.

A. Edge Detection

The first step involves dividing the cover image into small blocks, which are analyzed to classify them as edge or non-edge blocks. Edge detection is performed using the gradient magnitude method, which calculates pixel intensity changes between adjacent pixels in both horizontal and vertical directions.

Gradient Magnitude Calculation: For each block, the gradients in the x-direction g_x and y-direction g_y are computed as:

$$g_x = I(x+1, y) - I(x, y),$$
$$g_y = I(x, y+1) - I(x, y).$$

The gradient magnitude is then calculated using:

$$\text{gradient\_magnitude} = \sqrt{g_x^2 + g_y^2}$$

If the gradient magnitude exceeds a predefined threshold, the block is classified as an edge block, indicating regions with high contrast (e.g., object boundaries). Non-edge blocks, with lower contrast, are less perceptible to changes.

B. Text Message Conversion to Binary

The secret text message is converted into a binary string for embedding. Each character is transformed into its 8-bit ASCII binary representation using the following steps:

1) The ASCII value of each character is obtained using the ord() function.
2) The ASCII value is converted to an 8-bit binary format.

For example, the character 'A' is converted as follows:

'A'→ASCII(65)→Binary(01000001)

C. Image Conversion for Embedding

When embedding an image into another image, the hidden image undergoes preprocessing to convert it into a binary sequence. This sequence serves as the data to be hidden within the cover image.

1) Loading the Hidden Image:: The hidden image is first loaded and converted into RGB format. This ensures the image can be uniformly processed, with each pixel represented by its red, green, and blue (RGB) intensity values.

2) Flattening and Binary Conversion:: The pixel values of the hidden image are then flattened into a one-dimensional sequence. Each pixel intensity, ranging from 0 to 255, is converted into its binary representation using 8 bits per channel. This step results in a continuous binary sequence that encapsulates the entire content of the hidden image.

3) Preparation for Embedding:: The binary sequence is stored as a string, ready for embedding. To ensure compatibility with the cover image, the size of the binary sequence is calculated in advance. This helps determine whether the entire hidden image can be embedded or if adjustments (e.g., resizing or compressing the hidden image) are needed.

4) Alignment with the Cover Image:: Embedding is performed block-by-block within the cover image. Each block corresponds to a segment of the binary sequence. The number of bits embedded in a block depends on whether it is classified as an edge or non-edge block. Edge blocks, having higher capacity due to their naturally higher contrast, can accommodate more bits. Non-edge blocks are used sparingly to minimize perceptual distortion.

This process ensures that the hidden image is optimally converted and aligned with the embedding algorithm. By treating the hidden image as a continuous binary stream, the method enables efficient and imperceptible embedding within the cover image while maintaining flexibility for various image sizes and complexities.

D. Text Extraction

The extraction process involves reading the LSBs of the image pixels and reconstructing the binary sequence of the embedded message.

Bit Extraction The extract-bits() function reads the LSBs of each pixel in the same order as they were embedded. The extracted bits are grouped into 8-bit chunks. Message Reconstruction The binary sequence is converted back to characters:

Each 8-bit chunk is interpreted as an ASCII value. The ASCII value is transformed into its corresponding character using the chr() function.

E. Image Extraction

For extracting an embedded image:

1) The extracted binary sequence is divided into 8-bit groups.
2) Each 8-bit group is converted back to pixel values.
3) The pixel values are reshaped into the dimensions of the hidden image.

The reshaped array forms the hidden image, which is then saved or displayed.

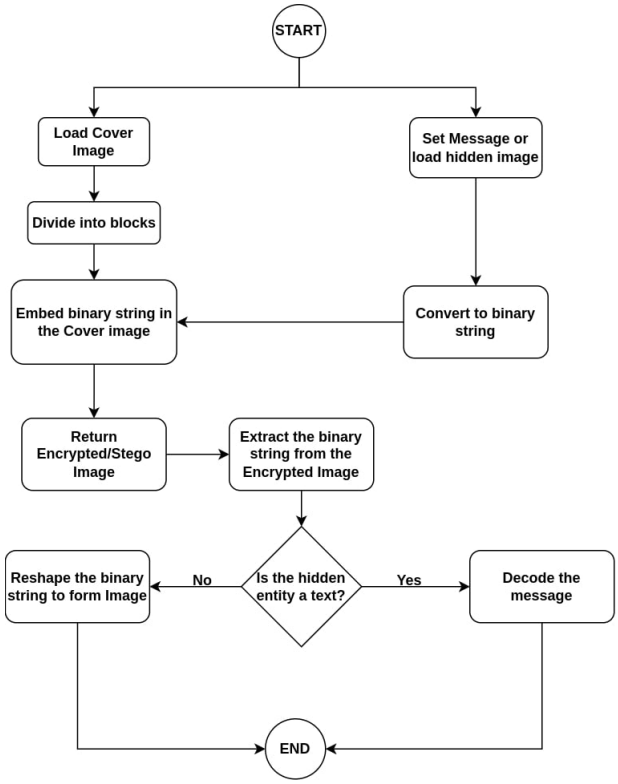

Fig. 1. Methodology Flowchart

IV. Results and Comparison

A. Hiding Text in Image

Figure 2 is the cover image inside which the below mentioned text is to be hidden. Text to be hidden: A journey of a thousand lines begins with a single idea.

Fig. 2. Cover image

Figure 3 is the stego image obtained which contains the embedded text.

Fig. 3. Stego image

Extracted text: A journey of a thousand lines begins with a single idea.

Histogram Analysis:

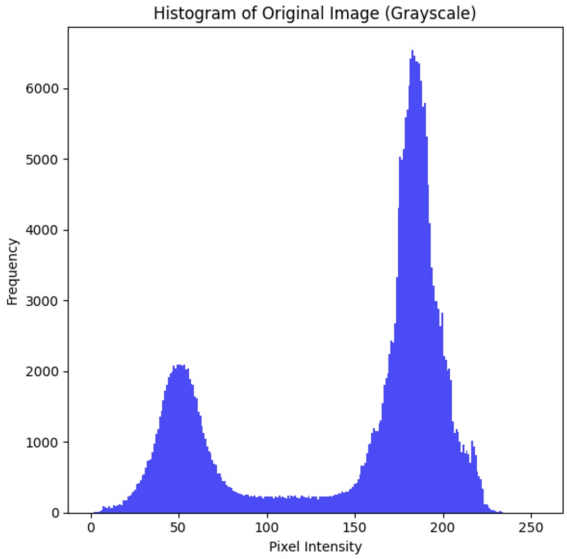

Fig. 4. Histogram of original image

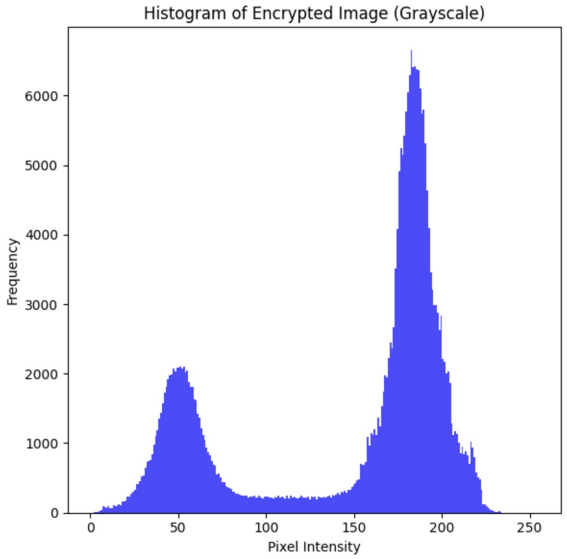

Fig. 5. Histogram of Stego image

1) Pixel Intensity Distribution:
   - The histogram of the original image shows a natural distribution of pixel intensities with clear peaks and valleys, reflecting the inherent characteristics of the cover image.
   - The histogram of the encrypted (stego) image appears slightly smoothed, with less pronounced peaks and valleys, indicating subtle modifications in the pixel intensities.
2) Small Deviations Across Intensity Ranges:
   - The encrypted image histogram may exhibit minor shifts or spikes in pixel frequency at certain intensity levels where the hidden data was embedded. These deviations suggest the presence of additional data that was not part of the original image.

- The jagged lines at the peaks and throughout the second peak of the graph suggest that the text has been embedded in that part of the image. This is analogous to the image, as the sky in the top part of the image lacks many edges. However, the text is hidden as soon as edges are detected.

Chi-Square Test Analysis:

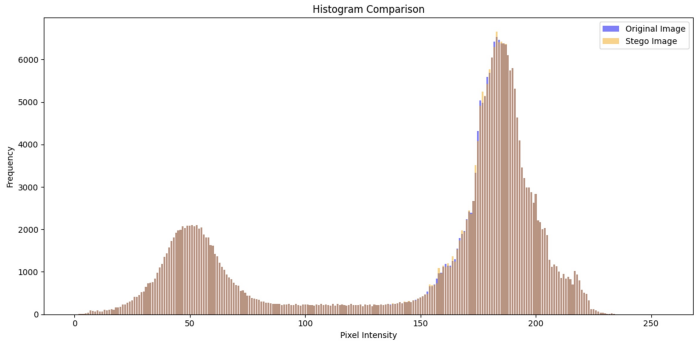

Fig. 6. Chi-Square Analysis

The chi-square analysis highlights the statistical deviations introduced by the embedding process. Significant peaks in the graph indicate regions where pixel intensity distributions were altered, while lower chi-square values suggest areas of minimal disruption. These variations demonstrate the trade-off between embedding capacity and imperceptibility. A well-designed steganographic algorithm aims to minimize these deviations, ensuring the stego image remains visually indistinguishable from the original while securely embedding the data. The results emphasize the importance of adaptive techniques to balance data hiding efficiency and detection resistance.

B. Hiding Image in Image

Figure 7 is the cover image inside which the secret image is to be hidden.

Fig. 7. Cover image - Image in which secret image is to be embedded

Figure 8 is the secret image which is to be hidden inside the cover image

Fig. 8. Secret image - Image to be hidden

Figure 9 is the stego image obtained which contains the embedded image.

Fig. 9. Stego image

Figure 10 is the extracted image obtained from decoding(removing) the stego image to obtain the secret image.

Fig. 10. Extracted image

Histogram Analysis:

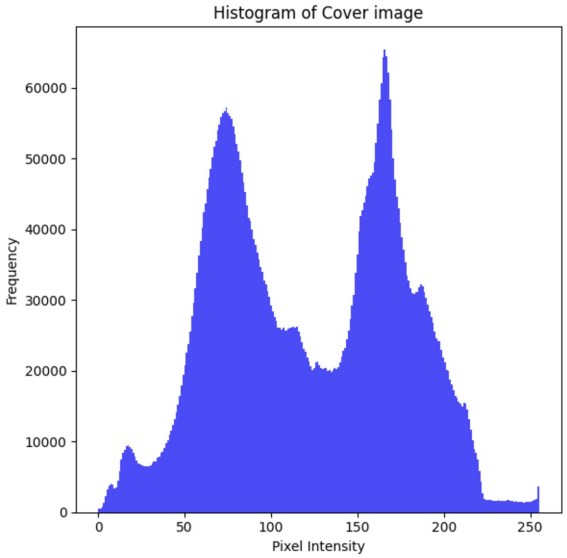

Fig. 11. Histogram of original image

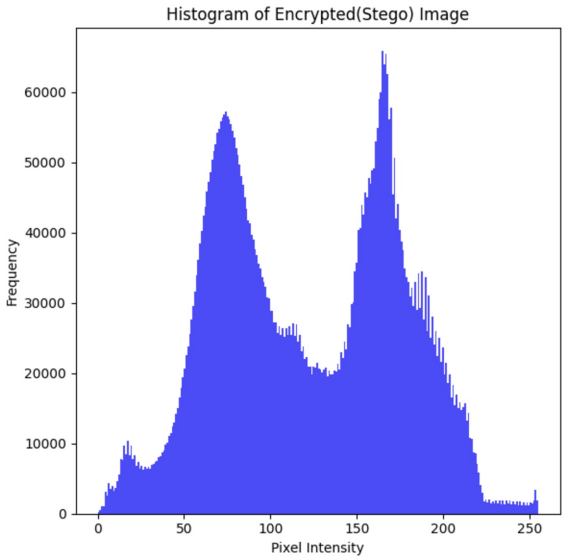

Fig. 12. Histogram of Stego image

- The histogram of the Stego image quite clearly has a lot more jagged lines and this is a clear indication of data, in this case an image, being embedded into the cover image.
- The concentration of more jagged lines in the second peak indicates the presence of more edges in the lower portion of the image which is true as the upper portion of the image mostly consists of the sky.

Chi-Square Test Analysis:

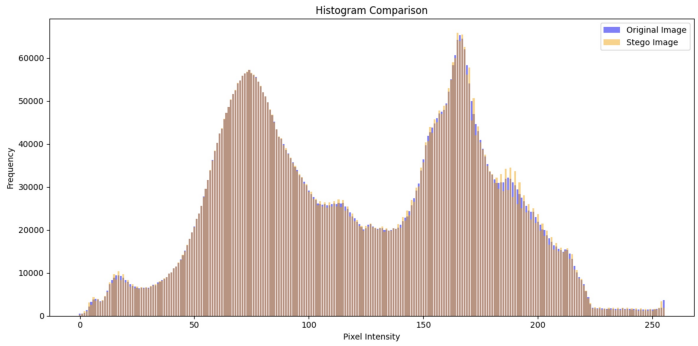

Fig. 13. Chi-Square Analysis

C. Metrics Evaluation

Metrics	SSIM	MSE	PSNR
Original Image	1	0	∞
Text-in-Image	0.999675	0.023974	64.333364
Image-in-Image	0.996590	0.305279	53.283832

TABLE I
COMPARISON OF IMAGE QUALITY METRICS FOR DIFFERENT EMBEDDING TECHNIQUES.

V. Acknowledgment

We would like to express our sincere gratitude to Dr. Janani ma'am for their invaluable guidance, constructive feedback, and unwavering support throughout this project. Their insights have been instrumental in shaping the direction of our research and ensuring the successful completion of this work.

References

[1] Hussain, Mehdi, et al. "Image steganography in spatial domain: A survey." Signal Processing: Image Communication 65 (2018): 46-66.
[2] Gaurav, Kumar, and Umesh Ghanekar. "Image steganography based on Canny edge detection, dilation operator and hybrid coding." Journal of Information Security and Applications 41 (2018): 41-51.
[3] Fateh, Mansoor, Mohsen Rezvani, and Yasser Irani. "A new method of coding for steganography based on LSB matching revisited." Security and communication networks 2021.1 (2021): 6610678.
[4] Ray, Biswarup, et al. "Image steganography using deep learning based edge detection." Multimedia Tools and Applications 80.24 (2021): 33475-33503.
[5] J. Hu, "Image Steganography based on improved LSB algorithm," 2024 39th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Dalian, China, 2024, pp. 1673-1677, doi: 10.1109/YAC63405.2024.10598763. keywords: Steganography;Automation;Accuracy;Measurement uncertainty;Copyright protection;Distortion;Indexes;Image Steganography;Optimization Model;Random Search Algorithm;LSB Algorithm