ECN-316: Digital Image Processing

Indian Institute of Technology Roorkee

Department of Electronics and Communication Engineering

Cryptic Images: Advanced Steganography and Steganalysis through Image Processing

Aryan Chaudhary
22115031
aryan_c@ece.iitr.ac.in

Nitesh Thalor
22116062
nitesh_t@ece.iitr.ac.in

Sarthak Shaurya
22116082
sarthak_s@ece.iitr.ac.in

Pandit Jwalit
22117099
pandit_jt@ece.iitr.ac.in

Date: November 29, 2024

Contents

Abstract

Steganography is the ancient art of concealing information within harmless mediums that might have undergone metamorphosis in the digital age. In this study, we try to investigate and implement an ensemble of steganographic methodologies, transcending both spatial and frequency domains, to explore their efficacy, robustness, and adaptability under diverse operational scenarios. The implemented algorithms span foundational techniques such as Least Significant Bit (LSB) manipulation and Discrete Cosine Transform (DCT) encoding, alongside approaches like edge-based steganography, histogram shifting, random pixel embedding, and frequency-domain transformations such as Fast Fourier Transform (FFT) and Discrete Wavelet Transform (DWT). The report concludes with a comparative analysis of all the discussed techniques. Their trade-offs are highlighted in terms of imperceptibility, payload capacity, and resilience against steganalysis.

1 Introduction

It is crucial to ensure the confidentiality and integrity of the transmitting data. When encrypted data is transmitted, robustness and security must be provided at two different levels. First of all, it should be difficult to detect that the image is encrypted, and secondly, if it is detected that the image is encrypted, then it must be difficult to decode the hidden information. Using the various steganographic algorithms, we try to hide the information within ordinary looking files like images, audio, or video. This is a brilliant approach to safeguarding the content and also concealing the presence of the message, adding an extra level of security.

There has been a massive increase in the use of multimedia in recent years, leading to a vast increase in the need for secure communication. This has necessitated the advancement of digital steganography. Some of the techniques, like LSB use minor and imperceptible changes in information for the purpose of information embedding, whereas some others, like the frequency domain approaches (Ex: DCT, DWT), take advantage of the way our eyes perceive changes in different frequency components of an image. Usually high frequency regions are used to embed information without bringing much distortion in the image.

More advanced approaches, like edge-based steganography, focus on embedding data in prominent areas of an image—those with edges or detailed features—where minor changes are less noticeable. Various other methods like histogram shifting and pixel value differencing, provide ways to improve imperceptibly as well as visual appeal of the encoded image with respect to the original image. In this study, we explore the several methods and compare the approaches on various metrices.

2 Problem Definition

There is a concomitant rise in the need for secure data transmission due to the proliferation of digital communication. The traditional cryptographic measures are easy to implement, but they are not secure enough, which means their security is unwarranted. Steganography addresses this lacuna with its promise of imperceptible communication by hiding data within digital carriers in a way that minimizes the probability of statistical detection. However, there are various challenges in designing a robust steganographic algorithm. Some of these are:

- **Imperceptibility vs. Capacity:** It is important to ensure that the image remains indiscernible to the human visual or auditory systems. However, there is a fundamental trade-off with maximising the payload capacity.
- **Robustness to Attacks:** The algorithms must be robust against unintentional and unrequited distortions. It should be resilient against compression, filtering and steganalysis attacks that try to obliterate hidden messages.
- **Algorithmic Complexity:** There should not be a large computational overhead associated with the algorithms.

In this project, we try to address these challenges by implementing and evaluating various algorithms, some of which operate in the spatial domain while some operate in the frequency domain. We aim to provide a granular understanding of their relative advantages and limitations.

Criterion	**LSB**	**DCT**	**Edge-Based**	**Hist. Shift.**
Domain	Spatial	Frequency	Spatial	Spatial
Imperceptibility	Low	High	High	Moderate
Payload Capacity	High	Moderate	Moderate	High
Robustness	Low	High	Moderate	Moderate
Comp. Complexity	Low	High	Moderate	Moderate
Resist. to Steganalysis	Low	Moderate	High	Moderate
Suit. for Compression	Poor	Good	Moderate	Moderate
Application Scenarios	Basic, high capacity	Media watermarking	Sensitive concealment	High capacity

Table 1: Comparative Analysis of Steganography Algorithms (Part 1: LSB, DCT, Edge-Based, Histogram Shifting)

Criterion	**Rand. Pixel**	**FFT**	**DWT**	**PVD**
Domain	Spatial	Frequency	Frequency	Spatial
Imperceptibility	Moderate	High	High	Moderate
Payload Capacity	Low	Low	Moderate	High
Robustness	Low	High	High	Moderate
Comp. Complexity	Low	High	High	Moderate
Resist. to Steganalysis	Moderate	High	High	Moderate
Suit. for Compression	Poor	Good	Good	Moderate
Application Scenarios	Low detection risk	Watermarking	Secure comms, compression	Image auth.

Table 2: Comparative Analysis of Steganography Algorithms (Part 2: Random Pixel Embedding, FFT, DWT, PVD)

3 Least Significant Bit (LSB) Steganography

Steganography has emerged as a vital tool for concealing sensitive information in the evolving field of data security and privacy. Cryptographic algorithms do ensure data security but fail to obscure the presence of data. Steganography embeds information in a way that renders its presence imperceptible.

LSB Steganography is a spatial domain steganographic algorithm that has emerged as one of the earliest and easiest approaches due to its simplicity. It also has a high payload capacity. The human visual system is less sensitive to subtle variations in pixel intensities because of minor changes in the original image. LSB, however, has some fundamental vulnerabilities. It is susceptible to statistical steganalysis and has weak robustness against lossy compression. It also has a high risk of detection in noisy environments.

3.1 Embedding Algorithm

1. Convert the secret message into its binary representation and append a null character (\0) to mark the end of the message.
2. Flatten the pixel values of the cover image into a 1D array.
3. For each bit of the binary message:
    (a) Modify the least significant bit (LSB) of the corresponding pixel's R-channel value to match the message bit.
4. Reshape the modified array back into the original image dimensions and save it as the stego image.

3.2 Extraction Algorithm

1. Flatten the pixel values of the stego image into a 1D array.
2. Extract the LSB of each pixel's R-channel value to reconstruct the binary message.
3. Group the binary bits into bytes (8 bits each) and convert them to characters.
4. Stop when the null character (\0) is encountered, indicating the end of the message.

4 Discrete Cosine Transform (DCT)

The concerns regarding data security and privacy have been amplified due to the exponential growth of digital communication in recent decades. Conventional cryptographic methods do safeguard the content but fail to mask the existence of the data itself. Thus they are a target for unauthorized attention. DCT is compatible with lossy compression formats as it is a cornerstone of image compression standards like JPEG. DCT allows for subtle changes in coefficients as it transforms image data from the spatial domain to the frequency domain before embedding the message. This results in imperceptible changes in the carrier image.

4.1 Embedding Algorithm

Steps

1. Convert the secret message into binary format and append a 16-bit delimiter for message termination.
2. Divide the grayscale image into 8×8 non-overlapping blocks.
3. For each block:
    (a) Apply the 2D Discrete Cosine Transform (DCT) to transform pixel intensities into the frequency domain.
    (b) Quantize the DCT coefficients using a standard quantization matrix.
    (c) Embed the message bits into the mid-frequency coefficient at position $(4, 4)$ by modifying its least significant bit (LSB).

(d) Dequantize the modified DCT coefficients.

(e) Reconstruct the spatial block using the Inverse Discrete Cosine Transform (IDCT).

4. Save the modified image as the stego image.

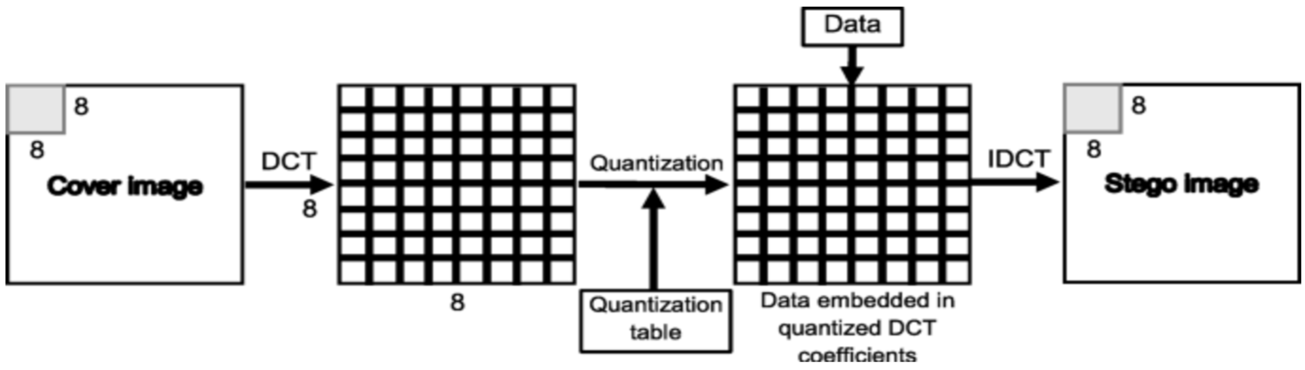

Figure 1: Encoding for DCT

4.2 Extraction Algorithm

Steps

1. Divide the stego image into 8×8 blocks.

2. For each block:

   (a) Apply the 2D Discrete Cosine Transform (DCT) to transform the block into the frequency domain.

   (b) Quantize the DCT coefficients using the same standard quantization matrix.

   (c) Extract the embedded bits from the LSB of the $(4, 4)$ coefficient.

3. Stop the extraction process upon detecting the 16-bit delimiter.

4. Convert the extracted binary sequence into the text message.

5 Edge-Based Steganography

Smooth regions are usually easier to detect in most of the traditional steganography algorithms. This issue is addressed by Edge-based steganography as it performs modifications in the less noticeable edge regions, which are characterized by abrupt intensity changes, where modifications are less noticeable. Edge detection algorithms like Canny, Sobel, or Prewitt are used in this technique to identify high-gradient areas that can be suitable for embedding. Data is localised to these regions, so this method has high imperceptibility and robustness against visual inspection. The statistical changes are less detectable by steganalysis tools in the edge regions. Thus, it is a preferred choice for applications intelligence and defence communications.

5.1 Embedding Algorithm

1. Convert the image to grayscale for edge detection.

2. Perform Canny Edge Detection to identify edge regions.

3. Extract the indices of edge pixels (non-zero intensity values).

4. Convert the secret message into binary format and append a delimiter (`1111111111111110`) to indicate the end of the message.

5. Check if the binary message can fit within the available edge pixels. If not, raise an error.

6. Embed the binary message into the least significant bits (LSBs) of the blue channel of the edge pixels.

7. Save the modified image as the stego image.

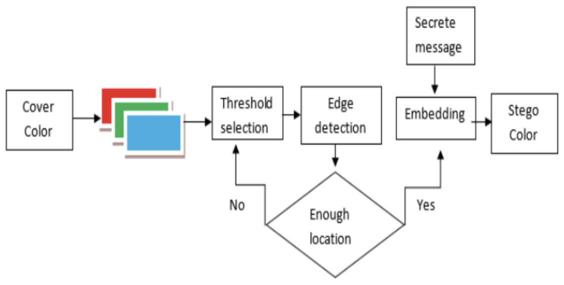

Figure 2: Encoding Algorithm

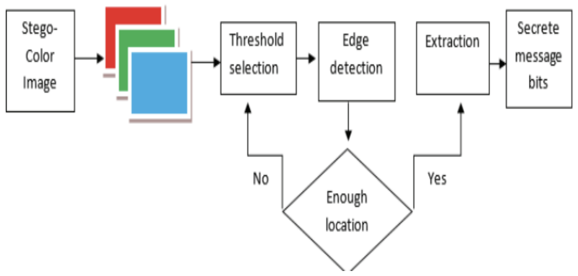

Figure 3: Decoding Algorithm

5.2 Extraction Algorithm

1. Convert the image to grayscale for edge detection.
2. Perform Canny Edge Detection to identify edge regions.
3. Extract the indices of edge pixels (non-zero intensity values).
4. Retrieve the binary data from the LSBs of the blue channel of the edge pixels.
5. Convert the binary data into 8-bit chunks and decode them into characters.
6. Stop the decoding process upon detecting the delimiter (`1111111111111110`).

6 Histogram Shifting Steganography

The early spatial domains usually introduce distortion and irreversibility challenges. The histogram shifting algorithm was introduced to overcome these. The statistical distribution of pixel intensities in an image are use in this technique identifies the peaks of the intensity histogram and shifts them to create space for embedding secret data. Histogram shifting minimizes perceptual distortion and ensures reversibility. The ability to embed data without significantly altering the cover medium makes this method particularly suitable for sensitive applications such as medical imaging, where image integrity is paramount. It is also resistant to lossy compression and thus, is more practical. However, the algorithm's embedding capacity is limited by the histogram characteristics of the cover image, necessitating careful image selection for optimal performance.

6.1 Embedding Algorithm

1. Convert the input image to PNG format if it is in JPEG format to avoid lossy compression.
2. Calculate the histogram of the image and identify the peak intensity (highest frequency) and a suitable zero point (intensity with low or zero frequency).
3. Shift the histogram to create a gap near the zero point without exceeding intensity bounds (0–255).
4. Convert the secret message to binary format and append a delimiter (`1111111111111110`) to indicate the end of the message.
5. Embed the binary message into the peak intensity pixels using the created gap.
6. Save the modified image as the stego image.

6.2 Extraction Algorithm

1. Convert the stego image to PNG format if it is in JPEG format to avoid lossy compression.
2. Calculate the histogram of the stego image and identify the peak intensity used during embedding.
3. Extract binary data from the modified peak intensity pixels.
4. Stop the extraction upon detecting the delimiter (`1111111111111110`).
5. Convert the extracted binary data into characters to retrieve the original message.

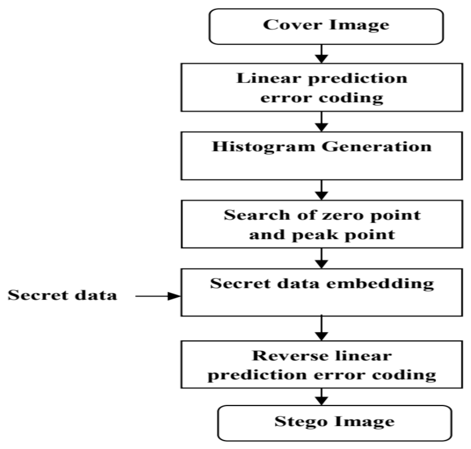
Cover Image
Linear prediction error coding
Histogram Generation
Search of zero point and peak point
Secret data
Secret data embedding
Reverse linear prediction error coding
Stego Image

Figure 4: Encoding Algorithm

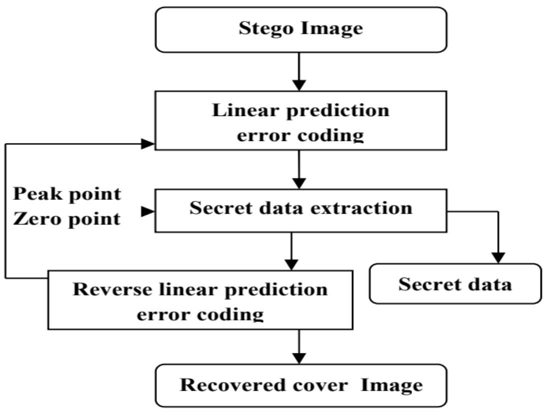

Figure 5: Decoding Algorithm

7 Random Pixel Embedding Steganography

The deterministic approaches got increasingly vulnerable as the sophistication of steganalysis algorithms increased. Random Value Embedding addresses this issue by introducing stochasticity into the embedding process by randomizing the sequence of pixels selected for embedding. Due to randomization, the detectability is minimized as the statistical anomalies that could trigger steganalysis are reduced. Pseudorandom Number Generators (PNGRs) or cryptographic algorithms are employed to guide pixel selection, ensuring that the hidden message is unpredictable and secure. This is highly important in high-security applications where the detection of algorithms might have critical consequences. The payload capacity is however reduced dyue to randomness.

7.1 Embedding Algorithm

1. Convert the secret message into a binary format.
2. Append a delimiter (`1111111111111110`) to the binary message to signify its end.
3. Obtain the dimensions of the cover image.
4. Generate a sequence of random pixel indices using a fixed seed for reproducibility.
5. Check if the binary message can fit within the total number of pixels.
6. For each bit in the binary message:
   (a) Use the random pixel index to determine the corresponding (x, y) coordinate.
   (b) Modify the least significant bit (LSB) of the blue channel of the selected pixel to store the message bit.
7. Save the modified image as the stego image.

7.2 Extraction Algorithm

1. Obtain the dimensions of the stego image.
2. Generate the same sequence of random pixel indices using the fixed seed.
3. Extract the LSBs from the blue channel of the pixels at the random coordinates.
4. Collect the extracted bits until the delimiter (`1111111111111110`) is detected.
5. Convert the extracted binary data back into characters to reconstruct the secret message.

8 Fast Fourier Transform Steganography

We need to make our steganographic algorithms capable of handling geometric distortions and transformations like rotation and scaling which led to the rise of Fast Fourier Transform (FFT) as a better choice than spatial domain techniques like LSB modification in steganographic techniques. High frequency components of the image are reflectance of edges in the image, where if we change anything slightly, cannot be visually detected by the human eye maintaining the hiding accuracy of the steganography algorithm. Also, Fast Fourier Transform's computational efficiency and its extensive use in signal processing applications made it a natural extension for steganographic purposes.

8.1 Embedding Algorithm

1. Convert the secret message into a binary format.
2. Append a delimiter (`1111111111111110`) to the binary message to signify its end.
3. Read the grayscale cover image and adjust its dimensions to be divisible by 8.
4. Divide the image into non-overlapping 8×8 blocks.
5. For each block:
    (a) Compute the FFT of the block.
    (b) Quantize the FFT coefficients using a predefined quantization matrix.
    (c) Embed one bit of the secret message into the least significant bit (LSB) of a mid-frequency FFT coefficient (e.g., $(4, 4)$).
    (d) Dequantize the modified coefficients and compute the inverse FFT to reconstruct the block.
6. Combine the modified blocks to form the stego image.
7. Save the stego image.

8.2 Extraction Algorithm

1. Read the stego image and adjust its dimensions to match the embedding process.
2. Divide the image into 8×8 blocks.
3. For each block:
    (a) Compute the FFT of the block.
    (b) Quantize the FFT coefficients using the predefined quantization matrix.
    (c) Extract the LSB of the selected mid-frequency FFT coefficient to recover one bit of the secret message.
4. Accumulate the extracted bits until the delimiter (`1111111111111110`) is detected.
5. Convert the binary data back into characters to reconstruct the secret message.

9 Discrete Wavelet Transform (DWT) Steganography

The limitations of spatial domain techniques, like their susceptibility to transformations like compression and noise, required a shift to the frequency domain. The Discrete Wavelet Transform (DWT) became a natural choice due to its inherent ability to provide multi-resolution analysis of images. DWT decomposes an image into different frequency sub-bands—low-low (LL), low-high (LH), high-low (HL), and high-high (HH)—each capturing distinct details of the image. Secret data is typically embedded into the higher-frequency bands (LH, HL, HH) where minor changes are less perceptible to human vision. Furthermore, DWT aligns with human visual system models, ensuring imperceptibility. The transform's scalability and resilience to compression and noise make it ideal for high-security applications like digital watermarking and secure communications, where robustness and imperceptibility are critical.

9.1 Embedding Algorithm

1. Convert the secret message into a binary format.
2. Append a delimiter (`1111111111111110`) to the binary message to signify its end.
3. Convert the cover image to a NumPy array.
4. Initialize an empty array for the encoded image.
5. For each color channel in the image:
    (a) Extract the current channel.
    (b) Perform the Discrete Wavelet Transform (DWT) on the channel data.
    (c) Quantize the wavelet coefficients using a predefined quantization matrix.
    (d) Embed one bit of the secret message into the least significant bit (LSB) of the quantized coefficients.
    (e) Dequantize the modified coefficients and reconstruct the channel using the inverse DWT.
6. Combine the modified channels to form the stego image.
7. Save the stego image.

9.2 Extraction Algorithm

1. Convert the stego image to a NumPy array.
2. Initialize an empty string to collect the extracted message bits.
3. For each color channel in the image:
   (a) Extract the current channel.
   (b) Perform the Discrete Wavelet Transform (DWT) on the channel data.
   (c) Quantize the wavelet coefficients using the predefined quantization matrix.
   (d) Extract the LSB of the quantized coefficients to recover the message bits.
4. Accumulate the extracted bits until the delimiter (`1111111111111110`) is detected.
5. Convert the binary data back into characters to reconstruct the secret message.

10 Pixel Value Differencing (PVD) Steganography

Pixel Value Differencing (PVD) arose from the need for adaptive embedding techniques that could balance payload capacity and imperceptibility. Unlike uniform methods that apply identical changes to all pixels, PVD exploits the variation in pixel intensity between neighbouring pixels to determine embedding capacity. We can incorporate more data in regions having high contrast (large intensity differences in adjacent pixel positions) without visual and perceptible distortion, while smoother regions are altered minimally to preserve visual quality. As smoother regions are low frequency (dc value) or more associated with the average value, which should not be altered. This approach tries to be compatible with the human visual system, making Pixel Value Differencing highly effective for high-capacity embedding scenarios. Its robustness and adaptability also make it resistant to many steganalysis attacks, ensuring its relevance in applications such as covert data transmission and digital forensics.

10.1 Embedding Algorithm

1. Convert the secret message into binary form.
2. Convert the cover image into three separate color channels (Red, Green, Blue).
3. For each color channel, process consecutive pixel pairs:
   (a) Calculate the difference between two consecutive pixels.
   (b) Find the optimal range for embedding and calculate the number of bits (`t`) that can be embedded in this difference.
   (c) If `t` is within the allowed range, embed the next `t` bits of the secret message into the pixel pair by modifying the difference.
   (d) Update the pixel values to store the embedded bits and clip them to the valid range (0-255).
4. Combine the modified color channels to form the stego image.
5. Save the stego image.

10.2 Extraction Algorithm

1. Convert the stego image into three separate color channels (Red, Green, Blue).
2. For each color channel, process consecutive pixel pairs:
   (a) Calculate the difference between two consecutive pixels.
   (b) Find the optimal range and calculate the number of bits (`t`) that were embedded in this difference.
   (c) Extract the embedded bits from the difference.
3. Accumulate the extracted bits and stop when all the data is recovered.
4. Convert the extracted binary data back to the original message by grouping the bits into bytes and converting each byte to its corresponding character.

11 Evaluation Metrics

Various metrics were used to evaluate the steganography techniques.

11.1 Peak Signal-to-Noise Ratio (PSNR)

PSNR measures the similarity between the cover and stego images. A higher PSNR indicates better image quality. It is calculated as:

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

where n is the maximum pixel value (typically 255 for an 8-bit image) and MSE is the Mean Squared Error.

11.2 Mean Squared Error (MSE)

MSE calculates the average squared difference between pixel values of the cover and stego images:

$$\text{MSE} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[J(i,j) - J'(i,j) \right]^2$$

where $J(i,j)$ and $J'(i,j)$ are the pixel intensities of the cover and stego images, respectively.

11.3 Structural Similarity Index (SSIM)

SSIM evaluates perceptual similarity considering luminance, contrast, and structure:

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

where μ_x, μ_y are the mean intensities, σ_x^2, σ_y^2 are variances, and σ_{xy} is the covariance.

11.4 Normalized Cross-Correlation (NCC)

NCC measures the similarity between the original and extracted secret images. It is calculated as:

$$\text{NCC} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} [C(i,j) - \mu_s][C'(i,j) - \mu_s']}{\sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (C(i,j) - \mu_c)^2 \sum_{i=1}^{M} \sum_{j=1}^{N} (C'(i,j) - \mu_c')^2}}$$

where $C(i,j)$ and $C'(i,j)$ are the pixel intensities of the cover and stego images, and μ_s, μ_s' are their means.

11.5 Payload Capacity

Payload capacity indicates the maximum amount of data embedded within the image:

$$C(\%) = \frac{\text{Pixels used for embedding}}{J(i,j)} \times 100$$

where $J(i,j)$ is the total number of pixels.

11.6 Accuracy

Accuracy measures the proportion of correctly recovered bits relative to the embedded message, indicating the reliability of data retrieval.

12 Simulation Results

The simulation was performed using 30 text messages of varying lengths and 20 grayscale images. Some example images are shown in the image below:

Figure 6: Cover Image 1

Figure 7: Cover Image 2

Figure 8: Cover Image 3

Figure 9: Cover Image 4

Figure 10: Cover Image 5

The message embedded for the measurements shown below was: ”The festival was alive with music, laughter, and joy, a celebration of community and the bonds they shared. As the clock struck midnight, they made wishes, believing that the magic of the moment would come true. He ventured into the forest, seeking solitude and inspiration among the towering trees and gentle rustling leaves.”

Table: Performance Metrics for Steganography Techniques

Cover Image	Cover Image Resolution	Technique	PSNR	SSIM	NCC	MSE	Payload Capacity	Accuracy (%)
1	510x489	DCT	33.88	0.8424	0.9968	26.56	1.0458	100
		DWT	37.70	0.9464	0.9955	11.05	1.0458	93.56
		Edge Based	inf	1	1	0	1.0458	100
		FFT	33.61	0.7770	0.9467	28.32	1.0458	100
		Histogram Shifting	48.70	0.9997	0.99999	0.88	1.0458	100
		LSB	75.22	1	1	0.002	1.0458	100
		Random Pixel Embedding	inf	1	1	0	1.0458	100
2	499x525	DCT	33.38	0.9144	0.9967	29.86	0.9955	90.80
		DWT	31.97	0.8669	0.9787	41.29	0.9955	85.58
		Edge Based	91.17	1	1	0	0.9955	62.88
		FFT	29.77	0.4865	0.8485	68.54	0.9955	93.25
		Histogram Shifting	41.44	0.9879	0.9996	4.67	0.9955	62.88
		LSB	76.95	1	1	0.0013	0.9955	100
		Random Pixel Embedding	90.85	1	1	0.00005	0.9955	62.88
3	494x517	DCT	33.52	0.8821	0.9958	28.94	1.0212	100
		DWT	34.44	0.9134	0.9915	23.41	1.0212	100
		Edge Based	inf	1	1	0	1.0212	100
		FFT	30.09	0.5127	0.9031	63.69	1.0212	100
		Histogram Shifting	48.68	0.9997	0.99999	0.88	1.0212	100
		LSB	75.81	1	1	0.0017	1.0212	100
		Random Pixel Embedding	inf	1	1	0	1.0212	100
4	859x481	DCT	35.78	0.9382	0.9983	17.18	0.6312	100
		DWT	31.76	0.8060	0.9643	43.37	0.6312	100
		Edge Based	inf	1	1	0	0.6312	100
		FFT	30.76	0.6806	0.9486	54.54	0.6312	100
		Histogram Shifting	48.35	0.9997	0.99999	0.95	0.6312	100
		LSB	77.99	1	1	0.001	0.6312	100
		Random Pixel Embedding	inf	1	1	0	0.6312	100
5	412x273	DCT	32.15	0.7479	0.9930	39.61	2.3187	66.26
		DWT	37.13	0.9049	0.9926	12.61	2.3187	100
		Edge Based	inf	1	1	0	2.3187	0.3067
		FFT	33.57	0.8349	0.9879	28.59	2.3187	66.26

Cover Image	**Cover Image Resolution**	**Technique**	**PSNR**	**SSIM**	**NCC**	**MSE**	**Payload Capacity**	**Accuracy (%)**
		Histogram Shifting	59.91	0.9996	0.99999	0.066	2.3187	0.3067
		LSB	72.06	1	1	0.004	2.3187	100
		Random Pixel Embedding	90.19	1	1	0.00006	2.3187	0.3067

13 Conclusion

In this study, we used various algorithms including several spatial and several transform domain methods. Each technique was evaluated based on various metrics like SSIM, PSNR, NCC, MSE, Payload Capacity, and accuracy.

The results show that frequency domain techniques are often better in terms of robustness and are suitable for scenarios that demand high reliability. Spatial domain methods are suitable where more embedding capacity is required. Histogram Shifting and PVD consistently achieved high PSNR and SSIM, showing their potential for high-quality image steganography. Edge-based method and DWT excellently balanced imperceptibility and robustness.

No single technique is better than others on all performance metrics. Thus, it is important to align steganographic strategies with the specific demands of the application to leverage their respective advantages.

References

[1] Alaa Almaliki, Farah Alyousuf, and Roshidi Din, ”Analysis review on spatial and transform domain technique in digital steganography,” *Bulletin of Electrical Engineering and Informatics*, vol. 9, pp. 573–581, Apr. 2020, doi: 10.11591/eei.v9i2.2068.

[2] Chaithra I. V, Sunitha M. R, and Vivekananda, ”A reversible steganography using prediction value coding and histogram shifting,” in *2019 1st International Conference on Advances in Information Technology (ICAIT)*, 2019, pp. 534–539, doi: 10.1109/ICAIT47043.2019.8987322.

[3] Rasber Dh. Rashid and Taban F. Majeed, ”Edge Based Image Steganography: Problems and Solution,” in *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, 2019, pp. 1–5, doi: 10.1109/ICCSPA.2019.8713712.

[4] Nishant Madhukar Surse and Preetida Vinayakray-Jani, ”A comparative study on recent image steganography techniques based on DWT,” in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2017, pp. 1308–1314, doi: 10.1109/WiSPNET.2017.8299975.

[5] Ntivuguruzwa Jean De La Croix, Tohari Ahmad, and Fengling Han, ”Comprehensive survey on image steganalysis using deep learning,” *Array*, vol. 22, p. 100353, 2024, doi: 10.1016/j.array.2024.100353.

[6] ”Kaggle - ALASKA2 Image Steganalysis,” 2024. [Online]. Available: https://www.kaggle.com/c/alaska2-image-steganalysis.