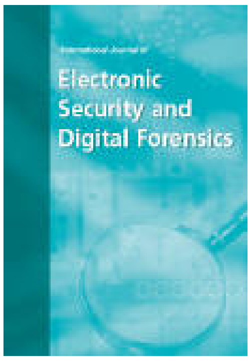


International Journal of Electronic Security and Digital Forensics

ISSN online: 1751-9128 - ISSN print: 1751-911X
https://www.inderscience.com/ijesdf

Exploring advanced steganography techniques for secure digital image communication: a comparative analysis and performance evaluation

Rohit Deval, Nachiket Gupte, Johann Kyle Pinto, Adwaita Raj Modak, Akshat Verma, Anirudh Sharma, S.P. Raja

DOI: 10.1504/IJESDF.2025.10058707

Article History:

Received:	17 April 2023
Last revised:	04 July 2023
Accepted:	13 July 2023
Published online:	23 December 2024

Exploring advanced steganography techniques for secure digital image communication: a comparative analysis and performance evaluation

Rohit Deval*, Nachiket Gupte,
Johann Kyle Pinto, Adwaita Raj Modak,
Akshat Verma, Anirudh Sharma and S.P. Raja

School of Computer Science and Engineering,
Vellore Institute of Technology,
Vellore, Tamilnadu, India
Email: rohitketan.deval2020@vitstudent.ac.in
Email: nachiket.gupte2020@vitstudent.ac.in
Email: johannkyle.pinto2020@vitstudent.ac.in
Email: adwaitaraj.modak2020@vitstudent.ac.in
Email: akshat.verma2020a@vitstudent.ac.in
Email: anirudh.sharma2020@vitstudent.ac.in
Email: avemariaraja@gmail.com
*Corresponding author

Abstract: This is a digital age. In a world where everything seems to be public, privacy and confidentiality have never been more important. So, the combination of this aspect of our life and this need of our age is the ability to securely hide data in the digital world in a way where it is not so easy to detect. Thus, the culmination of this thought process helped the authors arrive at the topic of our paper, which is steganography in digital images. Image steganography is defined as the process of 'concealing a message or piece of data inside an image file'. Image steganography is crucial in the digital era, when the transmission and storage of digital information are widespread, for protecting the confidentiality and integrity of sensitive data. To this end, it has been reviewed in the latest technology and has attempted to put forth the best techniques/algorithms by which data of many kinds can be hidden in digital images. After extensive research, it narrowed down to six techniques, which are presented in this paper.

Keywords: steganography; digital images; data hiding; encryption; secret message; least significant bit; LSB; steganography; image processing; image compression; image steganalysis; image quality metrics.

Reference to this paper should be made as follows: Deval, R., Gupte, N., Pinto, J.K., Modak, A.R., Verma, A., Sharma, A. and Raja, S.P. (2025) 'Exploring advanced steganography techniques for secure digital image communication: a comparative analysis and performance evaluation', *Int. J. Electronic Security and Digital Forensics*, Vol. 17, Nos. 1/2, pp.233–266.

Biographical notes: Rohit Deval is a BTech student and researcher at Vellore Institute of Technology, He has published papers on artificial intelligence in healthcare, and stock market prediction using machine learning techniques, and he is also a reviewer in easy-chair publications. His works include machine learning, deep learning, artificial intelligence, and cybersecurity.

Nachiket Gupte is currently pushing for a Bachelor of Technology degree in Computer Science at the esteemed Vellore Institute of Technology. In addition, he has a strong interest in machine learning and problem-solving and is constantly keen to explore new ideas and techniques in this fascinating area.

Johann Kyle Pinto is a passion-driven computer science student at VIT Vellore. He is currently working in the field of artificial intelligence and cyber security. He works with ML/DL-based research projects.

Adwaita Raj Modak is an eager third-year BTech student at VIT Vellore studying Computer Science with an IoT focus. He is a member of the VIT Vellore Computer Science Society and competes in hackathons and coding contests. He understands device connectivity and communication thanks to his IoT focus.

Akshat Verma is a BTech student and researcher at Vellore Institute of Technology. He loves applying technology to real-world challenges, especially in sustainable development. He thinks technology may help solve climate change and poverty. He wants to use IoT to build smart cities and improve urban living. He is driven, passionate, and talented, making him a promising computer science and IoT professional.

Anirudh Sharma is a student at VIT University, Vellore, pursuing a Bachelor of Technology in Computer Science Engineering. He has a keen interest in the field of machine learning and its applications in various domains. He is a curious learner and always seeks new challenges and opportunities to enhance his knowledge and skills. He has also completed various online courses and workshops on machine learning and data science.

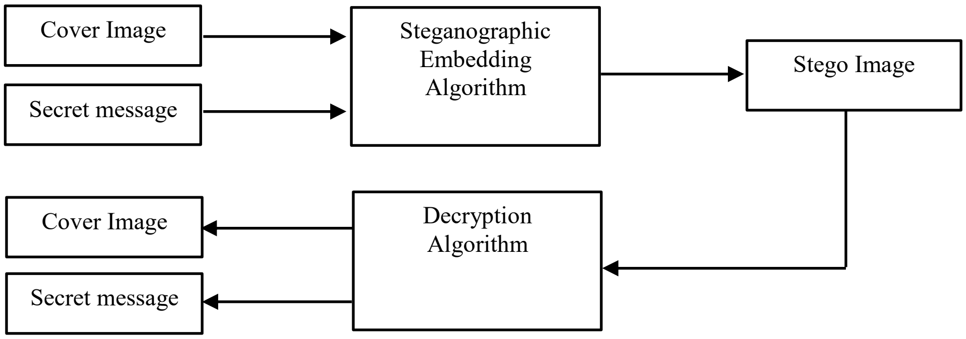
S.P. Raja completed his BTech in Information Technology in 2007 from Dr. Sivanthi Aditanar College of Engineering, Tiruchendur. He completed his ME in Computer Science and Engineering in 2010 from Manonmaniam Sundaranar University, Tirunelveli. He completed his PhD in 2016 from Manonmaniam Sundaranar University, Tirunelveli. His area of interest is image processing and cryptography. He is having more than 14 years of teaching experience. He has published 7 papers in international journals, 25 in international conferences and 12 in national conferences.

1 Introduction

Steganography, in general, refers to a method of concealing information within digital content like images, audio files, or videos. Generally, in encryption techniques it can be known whether an intercepted file is encoded even if it cannot be cracked. Steganography however ensures that the concealed information stays hidden even if the cover file is intercepted by others. The primary purpose of steganography is to maintain the privacy and confidentiality of the hidden data, which is crucial in fields such as banking, military, e-commerce and intelligence. Figure 1 shows the basic representation of steganographic image transmission:

The technique of steganography has always been around. From ancient times when messages were hidden in wax tablets with invisible ink to the modern world, where with the growing use of digital media and digital communication channels the need for steganography is greater than ever before. Present day digital steganography techniques are more advanced and sophisticated which makes it incredibly difficult to decode and extract information.

Figure 1 Steganography process



In this paper, the authors have explored various steganography methods and applied them to hide text and images within other images. The authors have successfully implemented six algorithms:

- least significant bit (LSB) substitution (text and image within image)
- pixel value difference 2× substitution
- pixel value difference 4× substitution
- LSB using edge detection
- reversible discrete cosine transform
- characteristic region based image steganography.

These techniques utilise various principles such as changing LSB of a pixel, altering pixel values, using edge detection, image transformation using DCT to hide secret messages.

One of the primary benefits of steganography over traditional encryption techniques is that the hidden data is difficult to detect. This is because the hidden data is embedded within the cover file, which remains unaltered to the naked eye. The hidden data can only be accessed by the intended recipient using a decryption key. Furthermore, steganography can be combined with encryption to provide an additional layer of security, making it nearly impossible for unauthorised users to access the concealed information.

The authors have studied recent papers to get a better understanding of current trends and state-of-the-art methods to get a better understanding of the current world of steganography.

Steganography has diverse and significant applications in various fields, such as banking, military, e-commerce and intelligence, for securing confidential information. It can also be utilised in digital forensics to retrieve hidden data and employed to protect privacy in personal communication, such as email or instant messaging.

In conclusion, steganography is a powerful tool for secure communication, and its applications are broad and significant in various fields. In this paper, the authors aim to demonstrate the implementation and importance of steganography algorithms in real-world scenarios and enhance their overall efficiency and applicability.

2 Related works

Evsutin et al. (2020) provides an overview of the current state of research in the areas of digital steganography for digital images. The authors review and analyse various existing techniques for both steganography and watermarking, highlighting their strengths and weaknesses, and identify potential areas for future research. The paper covers a range of topics including different types of steganography and watermarking, the role of human visual perception in digital image hiding, the use of machine learning and deep learning in steganalysis, and the challenges of developing robust and secure techniques for digital image hiding. Overall, the study is a valuable resource for academics and industry professionals involved in the field of digital image security.

Rahman et al. (2022) proposes a steganography technique that uses the LSB substitution method to embed secret data in digital images. The authors claim that their method provides better security and image quality than existing LSB-based techniques, and is able to withstand various attacks such as histogram analysis and statistical analysis. The paper describes the proposed technique in detail, including the process of selecting the pixels for embedding and the method of calculating the embedding capacity. The authors also provide experimental results to support their claims, demonstrating the effectiveness of their technique in terms of security and image quality.

Eid et al. (2022) provides a comprehensive survey of the state-of-the-art methodologies in digital image steganalysis, which is the detection and analysis of hidden information in digital images using steganography techniques. The authors discuss various types of steganography methods and analyse their strengths and weaknesses in terms of their detectability by existing steganalysis techniques. The paper also highlights the future challenges in digital image steganalysis and presents potential solutions to overcome these challenges. Overall, the paper provides a useful guide for researchers and practitioners working in the field of digital image steganalysis.

In this paper, Kaur et al. (2022) present a systematic review of various computational image steganography approaches. The review discusses the many types of picture steganography methods, including hybrid, frequency, and spatial approaches. The authors present a thorough examination of the state-of-the-art techniques within each area as well as a discussion of the benefits and drawbacks of each category. They also suggest prospective topics for future study and point out the difficulties and limitations of the approaches now in use. Overall, this study offers a thorough summary of the state of the art in computational picture steganography and might be a helpful resource for academics and industry professionals.

Mandal et al. (2022) provide a comprehensive review of the research on digital image steganography from 1990 to 2014. They discuss the various steganographic techniques used in digital image hiding and highlight the strengths and limitations of each technique. The authors also analyse the performance of different steganographic methods in terms of capacity, imperceptibility, and robustness. They also review the existing methods for steganalysis, which is the process of detecting the presence of hidden data in an image. The review concludes by identifying some future directions for research in digital image steganography. The authors suggest that future work should focus on developing efficient algorithms for embedding high-capacity data into images with minimal distortion, improving the security of steganographic systems, and developing new steganalysis techniques that are capable of detecting advanced steganographic methods.

Dhawan and Gupta (2021) provide an overview of various data security techniques of steganography. The authors begin by providing an introduction to steganography, its history, and the need for it. They then proceed to discuss various steganographic techniques, such as image steganography, audio steganography, and video steganography. The paper also covers various steganography algorithms, including LSB, parity coding, and phase encoding. The authors then compare and analyse these algorithms based on various parameters, such as data payload, embedding capacity, and robustness against attacks. The paper concludes with a discussion on the future direction of steganography research and its potential applications.

3 Methodology

This research paper focuses of the various steganography techniques that were researched up thoroughly such that they can be used and implemented.

3.1 LSB substitution (Rahman et al., 2008)

Each image is composed of pixels, and each pixel has either two or four components, depending on whether the image is greyscale or colour. Each component in a pixel is typically eight bits or one byte in size, resulting in a pixel that can be considered a binary number of either two or four bytes in size. Changing the LSB of a binary number increments or decrements the number by one. Because the human eye cannot distinguish between two pixels whose binary values only differ at the LSB position, data can be hidden in an image using this algorithm.

LSB substitution is a sort of steganography that involves concealing a secret message in an image or text file's LSB s without affecting the original file's appearance.

In the case of text files, LSB substitution works by replacing the LSB s of each character in a text message with the bits of the secret message. Since text files are composed of characters, each character can be represented by a binary code consisting of eight bits. Thus, by replacing the LSB of each character with the bits of the secret message, the text message can be concealed without affecting the text's readability.

Advantages: LSB substitution is simple to implement. Beginners and experts may use it by changing the cover medium's least important bits with the secret message bits. LSB substitution may incorporate huge pictures or cover material, particularly high-resolution photographs. Only the least important bits are updated, therefore a lot of hidden data may be embedded without changing the cover media.

Limitations: LSB replacement is vulnerable to statistical analysis, particularly when used to certain cover media or using unsophisticated methods. LSB statistical characteristics of changed cover media may disclose concealed information. Fragility: compression, resizing, and format conversion may affect LSB replacement on the cover media. These activities may modify or erase the embedded message, making it unrecoverable.

Suggestions: To improve the payload capacity of LSB substitution algorithm, the following steps can be taken. Firstly, selective embedding wherein, instead of modifying all pixels indiscriminately, selectively specific pixels or regions can be chosen for embedding. For instance, identify regions with less visual importance or lower human attention sensitivity where modifications are less likely to be noticed. By targeting suitable areas, the payload capacity can be increased. Another method could be implemented by exploiting coloured channels. For coloured images, LSB substitution can be extended to multiple colour channels (e.g., red, green, blue). By distributing the hidden data across different channels the payload capacity can be increased.

3.1.1 Algorithm for embedding message in the image

The embedding process in LSB substitution involves six steps. First, the original text message and the secret message are loaded. Then, the secret message is converted into a bitstream. Next, the text message is scanned, and the LSB of each character is replaced with the next bit of the secret message. If there are still bits remaining in the secret message, the process returns to the second step. Once all the bits of the secret message have been embedded, the stego text message is created.

1 begin by loading the image and obtaining the message that the user wants to embed

2 convert the message into a stream of bits

3 start scanning pixels in a predetermined sequence, beginning from the top-left pixel in the image

4 extract the next bit from the message bitstream, and retrieve the next pixel from the sequence

5 alter the LSB of the pixel's binary value to match the extracted bit from the message

6 if there are still bits remaining in the message bitstream or pixels in the image to be scanned, then return to step 4.

3.1.2 Algorithm for extracting message from the image

The extraction process, on the other hand, involves loading the stego text message and retrieving the secret message. The process starts by scanning the stego text message and extracting the LSB of each character. These bits are then combined to form a bitstream that represents the secret message. Once the bitstream is complete, it is converted back into its original form to retrieve the secret message.

1 Load the image and obtain the length of the message from the user.

2 Initialise an empty bitstream.

3 Begin at the top-left pixel of the image and traverse pixels in the same predetermined order as the embedding process.

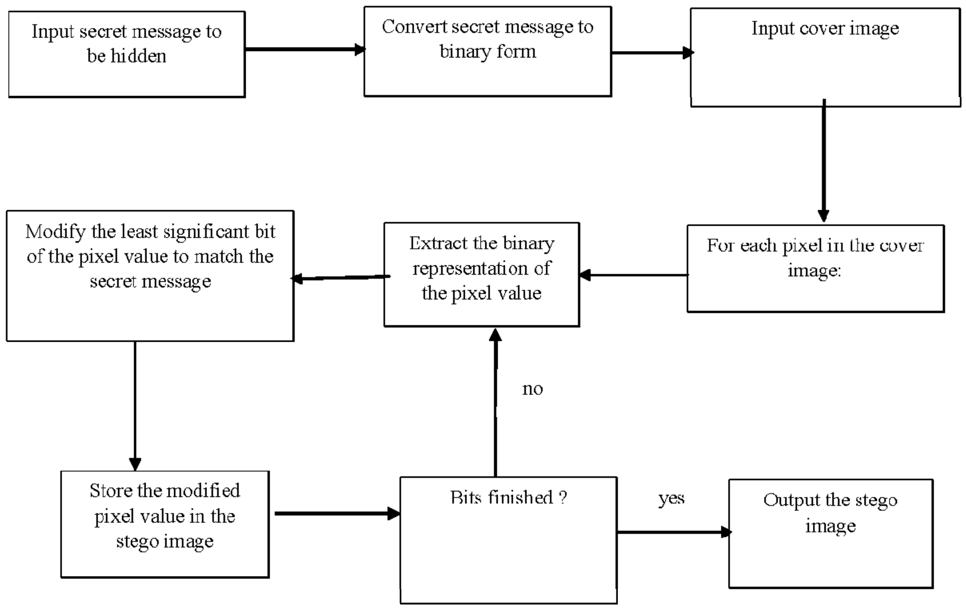
4 Obtain the next pixel from the sequence, read the LSB of the pixel's value, and append it to the bitstream.

5 If all pixels in the image have been scanned or the number of bytes in the bitstream equals the message length, proceed to step 6. Otherwise, repeat step 4.

6 Convert the bitstream to a string and output the recovered message.

One of the advantages of LSB substitution is that it is relatively easy to implement and can be used to hide a large amount of data in a text file without affecting its size. However, it also has some limitations, such as being vulnerable to statistical analysis, which can detect the presence of a hidden message. Additionally, LSB substitution is not suitable for images that require high levels of compression, as it can cause distortion and loss of data.

Figure 2 Architecture of LSB substitution method


Input secret message to be hidden
Convert secret message to binary form
Input cover image
For each pixel in the cover image:
Extract the binary representation of the pixel value
Modify the least significant bit of the pixel value to match the secret message
Store the modified pixel value in the stego image
Bits finished ?
no
yes
Output the stego image

3.1.3 Pseudocode for LSB substitution

```
// Inputs:
// – an image (as a matrix of pixel values)
// – a binary message to be embedded in the image
// Output: a modified image that includes the embedded message

// For each pixel in the image
for each pixel in image:
    // Convert the pixel value to binary
    binary_value = to_binary(pixel)
    // Modify the Least Significant Bit (LSB) of the pixel
    modified_value = modify_LSB(binary_value, message)
    // Convert the modified binary value back to decimal
    modified_pixel = to_decimal(modified_value)
    // Replace the original pixel value with the modified value in the image
    replace_pixel(image, pixel, modified_pixel)
```

```
// Output the modified image
output(image)
```

3.2 Pixel value difference 2× substitution

Pixel value difference 2× substitution (PVD2) (Fridrich et al., 2003) is a steganographic method used to hide information in digital images. Unlike other steganographic techniques, PVD2 modifies the values of pairs of adjacent pixels in an image to hide the message. This method is based on the fact that the difference between the values of adjacent pixels is usually small and imperceptible to the human eye (Fridrich et al., 2003).

The PVD2 algorithm operates by taking a cover image and a secret message, which is then converted into a binary format. The algorithm then iterates through each pixel in the cover image and checks if it can be used to store a bit of the secret message. PVD2 uses pairs of adjacent pixels to store a single bit of the message. If the difference between the pixel values is even, then the bit is stored as 0. If the difference is odd, then the bit is stored as 1. This is done by adding or subtracting one from the value of the first pixel in the pair, depending on the value of the bit being stored.

To ensure that the message is hidden as securely as possible, PVD2 employs a pseudo-random number generator to determine the order in which pixels are examined. This makes it difficult for an attacker to guess which pixels have been used to store the message. Additionally, PVD2 can be applied multiple times to the same image, using different secret messages and different random seeds, to further increase the security of the hidden data.

Advantages: PVD 2× substitution frequently produces more undetectable alterations than LSB substitution. Instead of changing bits, considering pixel values may make cover medium changes less obvious to humans. Higher embedding capacity: Compared to LSB substitution, PVD 2× replacement can offer a fairly big capacity for embedding. By changing pairs of pixels based on how they are different from each other, it is possible to hide more secret information in the cover medium without making it hard to see or hear.

Limitations: PVD 2× substitution frequently produces more undetectable alterations than LSB substitution. Instead of changing bits, considering pixel values may make cover medium changes less obvious to humans. Higher embedding capacity: Compared to LSB substitution, PVD 2× replacement can offer a fairly big capacity for embedding. By changing pairs of pixels based on how they are different from each other, it is possible to hide more secret information in the cover medium without making it hard to see or hear.

3.2.1 Algorithm for encoding

1 Load the cover image and the message to be embedded.

2 Convert the message into a bit stream.

3 Traverse the cover image, two pixels at a time, in a zigzag pattern.

4 For each pair of pixels, compute the absolute difference between their values.

5 Determine the number of bits to embed based on the difference value according to the embedding rules:

   a If the difference is less than or equal to 1, embed 1 bit.
   b If the difference is greater than 1 but less than or equal to 3, embed 2 bits.
   c If the difference is greater than 3 but less than or equal to 7, embed 3 bits.
   d If the difference is greater than 7 but less than or equal to 15, embed 4 bits.
   e If the difference is greater than 15, embed 5 bits.

6 Extract the required number of bits from the message bit stream.

7 Embed the extracted bits into the pixel pair according to the embedding rules:
   a If the difference is even, add the extracted bits to both pixel values.
   b If the difference is odd, add the extracted bits to the pixel with the smaller value.

8 Move to the next pair of pixels and repeat steps 4–7 until all message bits are embedded or all pixels in the cover image are scanned.

9 If there are any remaining message bits, pad the bit stream with zeros and embed them into the last pixel pair.

10 Output the stego image.

3.2.2 Algorithm for decoding

1. Load the stego-image.
2. Traverse the stego-image two pixels at a time, in the same zigzag pattern used during encoding.
3. Find the difference between two adjacent pixels.
4. Depending on the difference, recover the number of bits that were embedded during encoding.
5. Using the recovery scheme given in the encoding algorithm, find the bits of the message.
6. Add these bits to a bitstream.
7. If all the pixels in the stego-image are scanned or the number of bits in the bitstream equals the message length, move to step 8. Otherwise, return to step 3.
8. Convert the bitstream to a string and output the message.

3.2.3 Detection mechanism for the decoding algorithm

- PVD 2× substitution modifies pixels at edges, which have high contrast. Edge detection methods or edge maps may reveal modified areas in the cover picture. Edge discontinuities may suggest buried data.
- PVD 2× replacement alters cover picture pixel discrepancies. Pixel discrepancies reveal strange patterns. Compared to predicted statistical distributions, considerable deviations reveal hidden information.
- PVD 2× substitution may be detected by statistically comparing pixel values. Analysing the histogram of pixel differences may reveal odd patterns or statistical abnormalities. Chi-square testing, correlation analysis, and entropy calculations may uncover anomalous statistical areas.

One of the main advantages of PVD2 is that it does not significantly alter the cover image, making it difficult for an observer to detect that a message has been hidden in it. However, PVD2 is not without its limitations. One of the biggest drawbacks of this method is that it can be vulnerable to attacks that involve manipulating or altering the cover image. The concealed message may be compromised or lost if an attacker is able to alter the picture in a manner that changes the pixel values.

In conclusion, PVD2 is a steganographic method that offers a balance between imperceptibility and security. It is an efficient and simple algorithm that can be used to hide messages in digital images while maintaining the visual quality of the cover image. However, it is important to keep in mind that PVD2 is not completely fool proof and should be used in conjunction with other security measures to protect sensitive data.

3.3 Pixel value difference 4× substitution (Liao et al., 2011)

Pixel value difference 4× substitution is a steganographic technique used to hide information in digital images (Liao et al., 2011). This technique is an extension of the pixel value difference 2× substitution technique and is used to embed more data in an image.

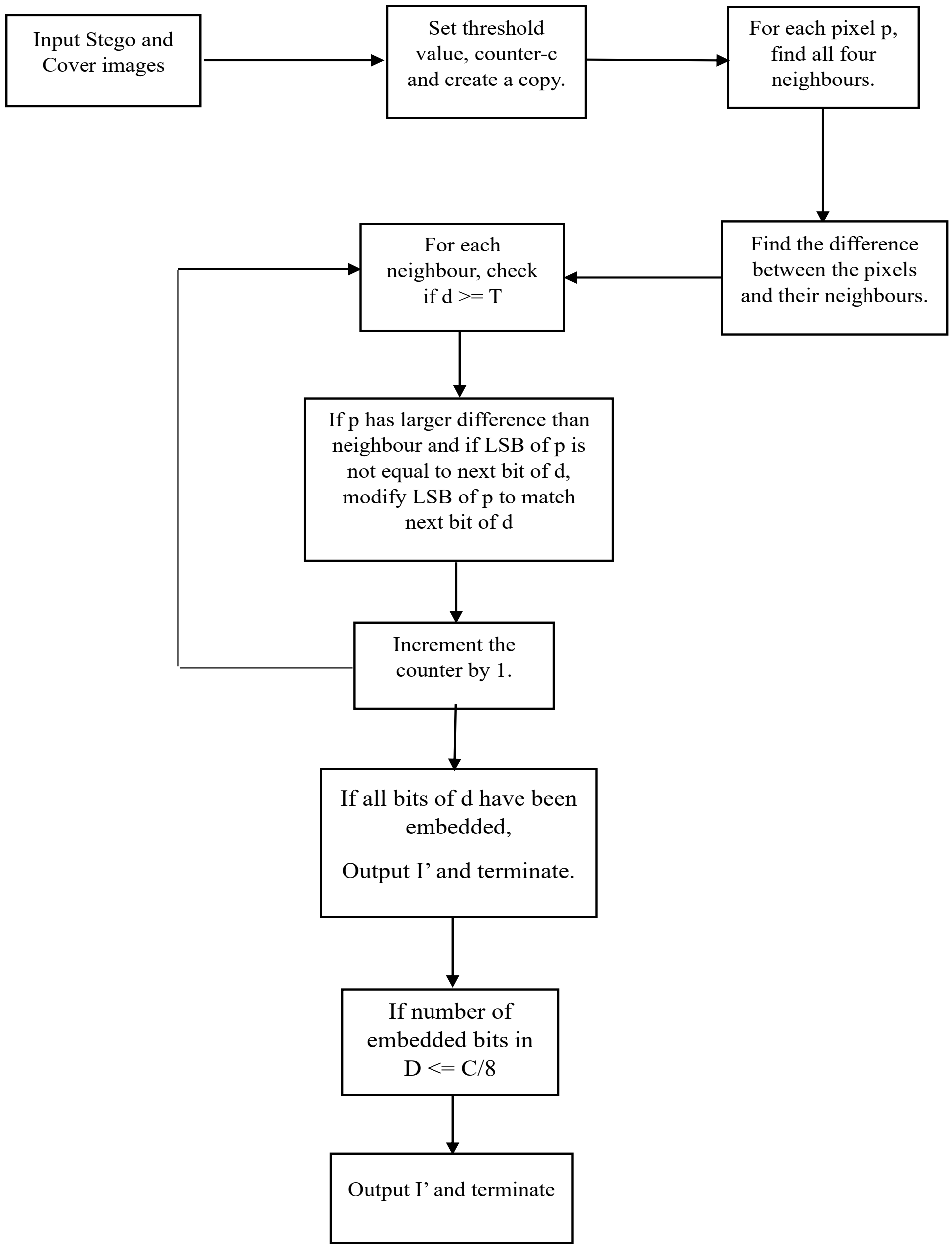
In this technique, each pixel value is represented by an 8-bit binary number. The difference between the two consecutive pixels in a pixel pair is computed, and based on this difference, a certain number of bits are embedded into the LSBs of the pixel values. The number of bits to be embedded is determined based on the magnitude of the difference between the two consecutive pixels. If the difference comes out to be larger, more bits are embedded.

Advantages: PVD 4× replacement is even more undetectable than PVD 2× substitution. Groups of four pixels may spread cover medium changes over a broader region, making them less visible to humans. PVD 4× substitution embeds more than PVD 2× and basic LSB substitution. By altering four pixels at once, the cover media may encode more hidden data without distortion.

Limitations: PVD 4× substitution requires more computations than PVD 2× or basic LSB replacement. Deriving pixel value differences and swapping groups of pixels involves processing resources and may increase encoding and decoding time. Fragility and robustness: PVD 4× substitution, like other embedding methods, is fragile when compressed, resized, or format converted. These alterations may modify or erase the encoded message, compromising its integrity and recoverability.

Suggestions: in order to increase the payload capacity of PVD 4× substitution the following measure can be applied. The PVD4× method operates in the wavelet domain, where different subbands represent different frequency components of the image. The higher-frequency subbands usually have more coefficients available for modification. These bands can be utilised for increasing the number of added coefficients thereby increasing the payload capacity of the algorithm. Another method could be to employ adaptive embedding techniques wherein, instead of relying on a fixed embedding strength or threshold, the adaptive method dynamically adjusts the embedding strength based on local properties of the image or the embedding capacity of each coefficient. Adaptive embedding can optimise the payload capacity in different regions of the image while minimising visual distortions.

Figure 3 Architecture of the proposed pixel value difference substitution method


Input Stego and Cover images
Set threshold value, counter-c and create a copy.
For each pixel p, find all four neighbours.
Find the difference between the pixels and their neighbours.
For each neighbour, check if d >= T
If p has larger difference than neighbour and if LSB of p is not equal to next bit of d, modify LSB of p to match next bit of d
Increment the counter by 1.
If all bits of d have been embedded,
Output I' and terminate.
If number of embedded bits in D <= C/8
Output I' and terminate

3.3.1 Algorithm for encoding

1 Load the cover image and the message to be hidden in the image.

2 Convert the message into a bitstream.

3 Traverse the image, four pixels at a time, in a zigzag pattern.

4 For each pixel pair, calculate the difference between the two consecutive pixels.

5 Determine the number of bits to be embedded based on the magnitude of the difference between the two pixels.

   a If the difference is less than or equal to 15, embed 2 bits.

   b If the difference is greater than 15 and less than or equal to 31, embed 3 bits.

   c If the difference is greater than 31 and less than or equal to 63, embed 4 bits.

   d If the difference is greater than 63 and less than or equal to 127, embed 5 bits.

   e If the difference is greater than 127 and less than or equal to 255, embed 6 bits.

6 Extract the required number of bits from the bitstream and embed them in the LSBs of the pixel values according to the following scheme:

   a For embedding 2 bits, change the last 2 bits of each pixel value.

   b For embedding 3 bits, change the last 3 bits of the first pixel value and the last 1 bit of the second pixel value.

   c For embedding 4 bits, change the last 4 bits of the first pixel value and the last 2 bits of the second pixel value.

   d For embedding 5 bits, change the last 5 bits of the first pixel value and the last 3 bits of the second pixel value.

   e For embedding 6 bits, change the last 6 bits of the first pixel value and the last 4 bits of the second pixel value.

7 If the bitstream is empty or all the pixels in the image have been processed, go to step 8. Otherwise go back to step 4.

8 Save the modified image as the stego-image.

3.3.2 Algorithm for decoding

1 Traverse the stego-image in a zigzag pattern.

2 For each group of four adjacent pixels, calculate the differences between them, i.e., subtract the values of the second pixel from the first, third from the second, and fourth from the third.

3 Determine the magnitude of the maximum and minimum differences in the group, and store them as max_diff and min_diff, respectively.

4 Calculate the average of the two middle pixels, and store it as avg_pixel.

5 Determine the bit capacity of the group as 2 * (max_diff – min_diff).

6 Extract the next n bits from the stego-image, where n is the bit capacity of the group.

7 Convert the extracted bits into an integer value, and add min_diff to it.

8 If the resulting value is greater than avg_pixel, subtract it from 255, otherwise leave it unchanged.

9 Assign the resulting value to the second and fourth pixels in the group.

10 If all the pixels in the stego-image are scanned, or the number of recovered bits is equal to the length of the secret message, output the message and stop. Otherwise, return to step 2.

3.3.3 Detection mechanism for the decoding algorithm

- PVD 4× substitution alters cover picture groups of four pixels. Analysing pixel value changes inside these blocks might reveal unusual patterns or aberrations. Statistical study of block differences or the distribution of differences may assist discover important changes.
- Pixel difference histograms may identify PVD 4× replacement. Unusual spikes or peaks in difference distributions may reveal hidden data. Histogram deviations from statistical expectations may indicate additional inquiry.
- PVD 4× replacement alters cover image structure. Textures, edges, and gradients may reveal changes in the picture. Edge detection, texture analysis, and structural similarity assessment may be used.

3.3.4 Pseudocode for pixel value difference substitution

```
// PVD algorithm for steganography
// Inputs:
// – I: an image (as a matrix of pixel values)
// – D: a binary message to be embedded in the image
// – T: a threshold value to determine significant differences
pvd_algorithm(image I, data D, T):
    // Get the dimensions of the image
    m, n = getDimensions(I)
    // Initialise counter variable to keep track of the number of embedded bits
    counter = 0
    // Create a copy of the image to store the modified pixel values
    I_copy = I.copy()
    // Loop through each pixel in the image
    for(int i = 0; i< m; i++):
        for(int j = 0; j < n; j++):
            // Get the value of the current pixel
            current_pixel_value = I[i,j]
            // Get the values of the four neighbouring pixels
            neighbours = getNeighbours(I, i, j)
            // Compute the difference between the current pixel and its neighbours
            diffs = neighbours – current_pixel_value
            // Loop through each difference value
            foreach(d:diffs):
                // Check if the difference value is significant enough
                if(abs(d) >= T):
                    // Check whether the next bit of the message matches
                    if(d > 0):
                        if( ((p & 1) != (d[c] & 1)) && counter < len(d)){
                            // Modify the pixel value
                            I_copy[i][j] ++;}


                    // Check whether the next bit of the message matches
                    else if( ((p & 1) != (D[C] & 1)) && counter < len(d)) {
                        // Modify the pixel value to embed the next bit
                        I_copy[i][j] --;
                        // Increment the counter variable
                        counter++;
                    }
                    // Check if all bits have been embedded successfully
                    if(counter >= len(d) ){
                        Return I_copy;
                    }
    getNeighbours(I, i, j):
        // Returns the four neighbours of the pixel
        m, n = getDimensions(I)
        left = I[i][max(0,j-1)] if j> 0 else 0
        right = I[i][min(n-1, j+1)] if j < n-1 else 0
        top = I[max(0, i-1)][j] if i > 0 else 0
        bottom = I[min(m-1, i+1)][j] if i < m-1 else 0
        return [left, right, top, bottom]
```

3.4 *Edge detect LSB substitution (Chen et al., 2010)*

In the previous two algorithms, the authors used the information in the local neighbourhood of the pixel. Next the authors analyse an algorithm developed by Chen et al. (2010), which looks at the global image data to determine the best pixels for embedding. It's known that pixels in the edge region of the image can tolerate a lot more distortion than in the smooth area without being perceptible.

Advantages: edge detect LSB substitution seeks to enhance imperceptibility by selectively modifying pixels in cover medium regions where visual changes are less likely to be detected by human observers (Chen et al., 2010). By concentrating on areas with prominent boundaries, which are already susceptible to variation, the modifications produced by this method are frequently less noticeable.

Limitations: edge detection is a computationally intensive operation; therefore, incorporating it into the embedding procedure adds complexity and time burden.

Especially for extensive cover media or real-time applications, the additional computational requirements may slow down the embedding and extraction processes.

Suggestions: one way to increase the payload capacity of this algorithm is to not use all the edge pixels but to selectively pick certain edge pixels which are less likely to be noticed visually. This not only makes our algorithm more robust but also ensures a satisfying image quality post encryption. Instead of replacing individual pixel LSBs, another method would be to use multi-pixel blocks along the detected edges. This approach allows for more efficient utilisation of the available edge pixels and increases the payload capacity.

3.4.1 Algorithm (embedding message in the image)

1 Load the image and get the message from the user.

2 Convert the message into bitstream.

3 Apply a Fuzzy edge detector to get the big edges in the image. Explained in Amarunnishad et al. (2008).

4 Apply the Canny edge detector to get the small edges in the image.

5 Divide the image in n pixel blocks [P1, …, Pn], n should not be more than 8. The value recommended by the authors is n = 3.

   a Find the status of the [P2, …, Pn] pixels. Given the set of pixels as [P2, P3, P4] and P2 and P3 are edge pixels and P4 is not then the status will be 110.

   b Embed this status value in P1.

   c From P2 onwards, for each pixel in the block embed k1-bits from the bitstream if the pixel is an edge pixels other embed k2-bits. Here k1 and k2 are values that are determined by trial and error basis. The authors of the algorithm recommend k2 = 1, 2 and k1 = 3, 4, 5.

6 If there are bits remaining the bitstream and pixels in the image, go to step 5 otherwise output the stego-image.

3.4.2 Algorithm (recovering message from the image)

1 Load the image and get the message length from the user.

2 Open an empty bitstream.

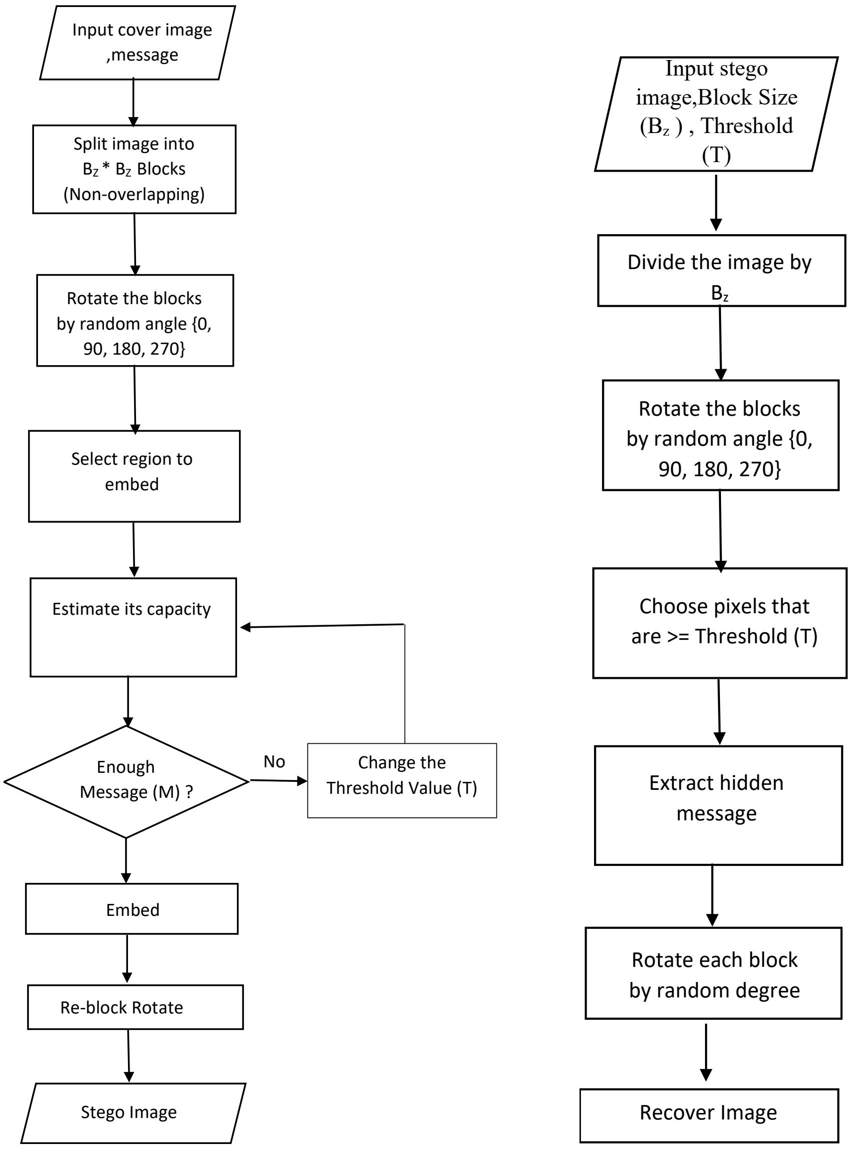
3 Traverse the image in sets of n pixels, the value of n here should be the same as used in the embedding process.

   a From the embedding process it is known that in this set of n pixels [P1, …, Pn], P1 contains the status of the set.

   b Extract the status of the set [P2, …, Pn] and depending on that extract bits from each pixel and insert them into the bitstream. Suppose the status extracted from P1 is 011 then k1 bits would be extracted from P3 and P4 and k2 bits from P2. Here k1 and k2 must have the same value as in the embedding process.

4 If the length of the bitstream is less than the length of the message and there are pixels remaining in the image, return to 3 otherwise output the message.

Figure 4 Architecture of proposed edge detect LSB substitution method


Input cover image ,message
Split image into $B_z * B_z$ Blocks (Non-overlapping)
Rotate the blocks by random angle {0, 90, 180, 270}
Select region to embed
Estimate its capacity
Enough Message (M) ?
No
Change the Threshold Value (T)
Embed
Re-block Rotate
Stego Image
Input stego image,Block Size (B_z) , Threshold (T)
Divide the image by B_z
Rotate the blocks by random angle {0, 90, 180, 270}
Choose pixels that are >= Threshold (T)
Extract hidden message
Rotate each block by random degree
Recover Image

3.4.3 Pseudocode for edge detect LSB substitution

```
input_image = read_image()
gray_image = convert_to_greyscale(input_image)
                                        threshold = 50
output_image = create_image_with_same_size_as(input_image)
```

```
for y in range(1, height-1):
    for x in range(1, width-1):
        current_pixel = gray_image[y, x]
        lsb_current = current_pixel & 1
        lsb_difference_sum = 0

        for I in range(-1, 2):
            for j in range(-1, 2):
                if I == 0 and j == 0:
                    continue

                neighbour_pixel = gray_image[y+I, x+j]
                lsb_neighbour = neighbour_pixel & 1
                lsb_difference_sum += abs(lsb_current – lsb_neighbour)

        if lsb_difference_sum > threshold:
          output_image[y, x] = 255 # white
        else:
          output_image[y, x] = 0 # black

return output_image
```

3.5 Reversible DCT image (Lin, 2012)

All the algorithms discussed so far operate in the spatial domain of the image, that is they all operate directly on the image. One disadvantage of using spatial domain is that algorithms in this domain are susceptible to image difference attack, that is if the attacker has the oral cover image, then he can easily determine whether any copy of that image has hidden data or not. To combat this, the authors highlight an algorithm that works in the frequency domain of the image. This algorithm was developed by Lin (2012). This algorithm also requires the image to be in the greyscale.

The DCT is mostly used in steganography processes that concentrate on concealing data inside picture or video files. It changes a signal or image’s frequency domain from the spatial domain. In steganography, hidden data is embedded by modifying an image’s DCT coefficients. Since the DCT coefficients’ LSBs are often more forgiving of changes, they may be swapped out for the secret data bits. The imperceptibility of the picture is preserved while encoding the concealed information by altering the LSBs.

The DWT is a steganographic method for inserting hidden information into many forms of digital media, such as pictures, audio files, and videos. A signal or a picture is divided into several frequency subbands using the DWT. Steganography involves changing the media’s DWT coefficients to hide the hidden information. Since they have less of an effect on the perceived quality of the media, the coefficients with greater frequency content are often adjusted to conceal the data.

Suggestions: The DWT decomposes an image into multiple subbands representing different frequency components. Higher-frequency subbands generally have more coefficients available for embedding. By focusing on these subbands, the payload capacity can be increased. Also, instead of embedding data in the coefficients of a single-level DWT decomposition, multi-level decomposition can be employed. This involves decomposing the image multiple times and embedding data in coefficients at different levels. Multi-level decomposition provides more coefficients for embedding, thereby increasing the payload capacity.

3.5.1 Algorithm (embedding message in the image)

1 Load the image and get the message from the user.

2 Convert the message into bitstream.

3 Each 8 × 8 block of pixels in the image can be converted into the frequency domain using the reversible factorisation of discrete cosine transform, described in Hao and Shi (2001). The process of converting each 8 × 8 block is described in Lin (2012).

4 Extract a bit from bitstream and embed in the block using the formula given below.

$$y'_{i,j} = \begin{cases} 2y_{i,j} & \text{if } -q<y_{i,j} < q \text{ and secret data is } 0 \\ 2y_{i,j}+1 & \text{if } -q<y_{i,j} < q \text{ and secret data is } 1 \\ Y_{i,j} & \text{if } q <= y_{i,j} \end{cases}$$

Here *p* and *q* are constants that depend on image, in general the author recommends p = 3 and p = 4.

5 Apply the inverse discrete cosine transform and replace the block in the original image.

6 If there are bits remaining the bitstream and pixels in the image, go to step 4 otherwise output the stego-image.

3.5.2 Algorithm (recovering message from the image)

1 Load the image and get the message length from the user.

2 Open an empty bitstream.

3 Each 8 × 8 block of pixels in the image can be converted into the frequency domain using the reversible factorisation of discrete cosine transform, described in Hao and Shi (2001). The process of converting each 8 × 8 block is described in Lin (2012).

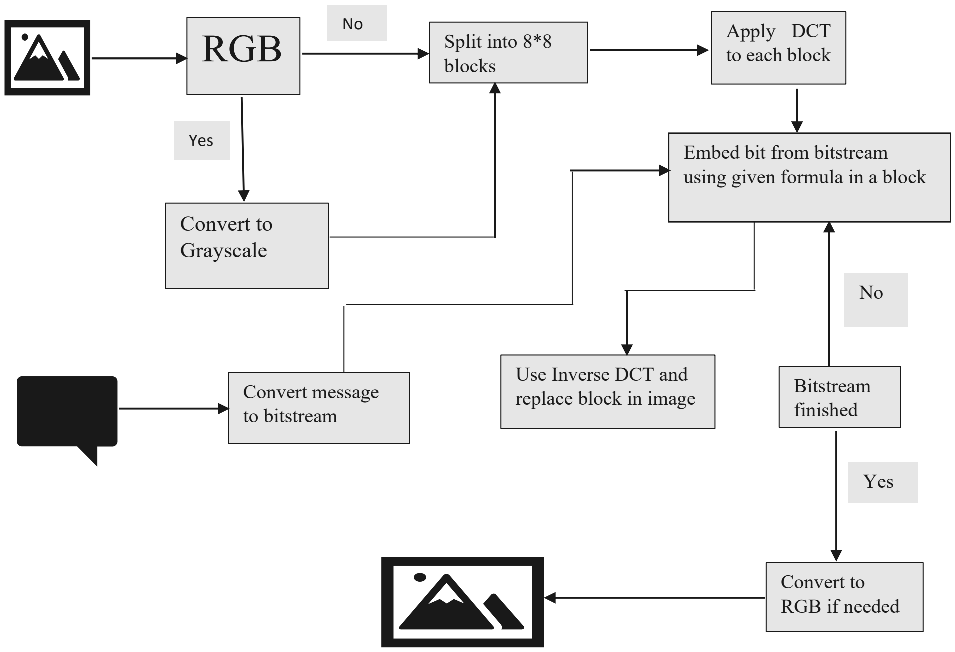
4 For each value *y*1 in the 8 × 8 frequency domain block obtained from the previous step.

   a If y1 is even, insert 0 in the bitstream.

   b If y1 is odd, insert 1 in the bitstream.

5 If the length of the bitstream is less than the length of the message and there are pixels remaining in the image, return to 3 otherwise output the message.

Figure 5 Architecture of proposed reversible DCT image method


RGB
No
Split into 8*8 blocks
Apply DCT to each block
Yes
Embed bit from bitstream using given formula in a block
Convert to Grayscale
No
Use Inverse DCT and replace block in image
Bitstream finished
Convert message to bitstream
Yes
Convert to RGB if needed

Note: In this case, the image generated will be RGB, but to depict in the diagram greyscale has been used.

3.5.3 Pseudocode for reversible DCT image substitution

```
// Inputs:
// – an image (as a matrix of pixel values)
// – a key value (used for embedding and extracting data)
// – a binary message to be embedded in the image
// – the block size (e.g. 8x8)
// Output: a modified image that includes the embedded message
// Define DCT matrix
DCT = create_DCT_matrix(block_size)
// For each block in the image
for each block in image:
    // Apply DCT to the block
    transformed_block = DCT * block * DCT’
    // Embed the message in the transformed block
```

```
        modified_block = embed_message(transformed_block, key, message)
        // Reverse the embedding process to extract the message
        extracted_message = extract_message(modified_block, key)
        // Reverse the DCT to obtain the original pixel values
        reverse_transformed_block = DCT’ * modified_block * DCT
        // Replace the original block with the modified block in the image
        replace_block(image, block, reverse_transformed_block)
    // Output the modified image
    output(image)
```

3.6 Characteristic region based image steganography

Previously developed steganography algorithms have addressed the challenges of making changes to an original image that are imperceptible, storing as much hidden data as possible, and protecting the stego image from steganalysis attacks. However, the issue of robustness still remains, which involves securing the hidden data against image transformations like rotation, cropping, and compression. In the case of steganography being used for transferring hidden data in images over the internet, image compression is unavoidable and can lead to loss of the hidden data. To overcome this problem, a sufficiently robust algorithm is needed, such as the characteristic region based image steganography (CR-BIS) developed by Yahya (2019). In CR-BIS, the feature detection algorithm SURF [developed by authors in Bay et al. (2018)] is used to detect features of the image that are likely to be preserved on compression, and the hidden data is then concealed in those areas. Discrete wavelet transforms are also used to safeguard the stego-image from basic steganalysis attacks. It is important to note that the image should be in greyscale for this algorithm to work properly.

3.6.1 Algorithm (embedding message in the image)

1 Load the image and obtain a message from the user.

2 Convert the message into a bitstream.

3 Detect feature points of the image using SURF and store them in a list.

4 Remove all points from the list except those that are separated from every other point by a distance of at least 2sqrt(2) * r, where r is a power of 2. The recommended value for r is 64.

5 Set a threshold value T to control the imperceptibility of the data.

6 For each remaining point in the list:

   a Choose a 2r * 2r square around the point.

   b Apply CDF 9/7 discrete wavelet transform to obtain four sub bands of the image: approximation (A), horizontal (H), vertical (V), and detail (D).

   c For each value H(x, y) and V(x, y) in the horizontal and vertical sub bands.

7 If $b = 1$ and $D1 = H(x, y) – V(x, y) < T$, modify $H(x, y)$ and $V(x, y)$ according to the provided formula.

8 $H'(x, y)$ is the modified value of the pixel at position (x, y) in the horizontal sub-band of the image.

$$H'(x, y) = H(x, y) + (T - D1)/2$$

$$V'(x, y) = V(x, y) - (T - D1)/2$$

9 If $b = 0$ and $D2 = V(x, y) – H(x, y) < T$, modify $H(x, y)$ and $V(x, y)$ according to the provided formula.

$$H'(x, y) = H(x, y) - (T - D2)/2$$

$$V'(x, y) = V(x, y) + (T - D2)/2$$

10 If there are still bits remaining in the bitstream and pixels in the image, return to step 6. Otherwise, output the stego-image.

3.6.2 Algorithm (extracting message from the image)

1 Load the image and obtain the length of the message from the user.

2 Create an empty bitstream to store the message.

3 Use SURF to detect feature points of the image and store them in a list.

4 Remove all points from the list except those that are separated from every other point with distance at least 2sqrt(2) * r, where r is a power of 2. The recommended value for r is 64.

5 For each remaining point in the list:

a Choose a 2r * 2r square around the point.

b Apply a CDF 9/7 discrete wavelet transform to obtain 4 sub bands of the image – approximation (A), horizontal (H), vertical (V), and detail (D).

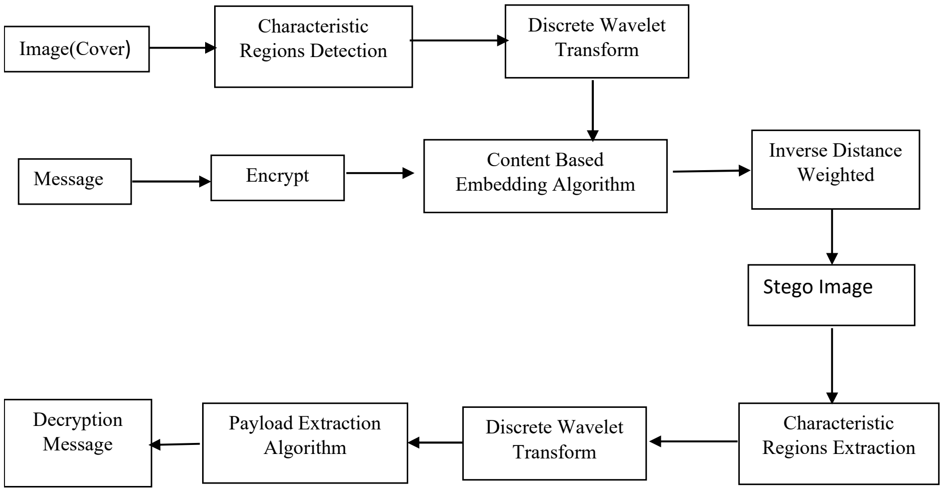
c For each value $H(x, y)$ and $V(x, y)$ in the horizontal and vertical sub bands, add a bit (represented as b) to the bitstream using the following value.

$$\text{If } H(x, y) > V(x, y), \text{ then set } b = 1$$

$$\text{If } H(x, y) < V(x, y), \text{ then set } b = 0$$

6 If the length of the bitstream is less than the length of the message and there are still pixels remaining in the image, return to step 5. Otherwise, output the message.

Figure 6 Architecture of proposed characteristic region based image steganography method


Image(Cover)
Characteristic Regions Detection
Discrete Wavelet Transform
Message
Encrypt
Content Based Embedding Algorithm
Inverse Distance Weighted
Stego Image
Decryption Message
Payload Extraction Algorithm
Discrete Wavelet Transform
Characteristic Regions Extraction

3.6.3 Pseudocode for characteristic region based image steganography

```
// Inputs:
// – an image (as a matrix of pixel values)
// – a binary message to be embedded in the image
// Output: a modified image that includes the embedded message
// Divide the image into characteristic regions
regions = divide_into_regions(image)
// For each region in the image
for each region in regions:
    // Calculate the region's statistical characteristics
    characteristics = calculate_characteristics(region)
    // Embed the message in the region's characteristics
    modified_characteristics = embed_message(characteristics, message)
    // Replace the original characteristics with the modified characteristics in the region
    replace_characteristics(region, modified_characteristics)
// Combine the modified regions to form the final modified image
modified_image = combine_regions(regions)
// Output the modified image
output(modified_image)
```

In steganography, embedding techniques are used to conceal confidential information within a cover medium, such as an image, audio file, or text, in a manner undetectable to human observers. The goal of steganography is to conceal the existence of a secret message, making it difficult for anyone other than the intended recipient to detect or recover the concealed data. Here are a few embedding techniques, discussed in brief.

3.7 Reversible embedding

Reversible embedding has really been the path that data concealing has taken in recent years. Reversibility indicates that the original cover object can be changed back to its original state once the embedded data has been extracted. This characteristic is especially important for applications involving the intelligent processing of images with embedded data, such as classification, pattern recognition, edge detection, etc. In this situation, even with a high degree of embedding imperceptibility, the additional information in the picture may lead to processing results that are skewed. The cover picture can only be restored to its original state by removing the imbedded data. Reversible embedding techniques can use both frequency coefficients and spatial data. Reversible embedding studies have been published for a while now.

Reversible embedding techniques and algorithms are typically used for steganography. In certain instances, though, we're referring to the reversible embedding of digital watermarks. A better reversible contrast mapping approach, for instance, is suggested in for reversible invisible watermarking. The spatial domain of the picture contains the fragile watermark.

Embedding in addition to secret sharing is a method that is now being actively developed for concealing information in digital photographs. These techniques' core tenet is that all information is distributed among numerous shadow pictures rather than being incorporated in a single image. As a consequence, a full message that only contains the necessary components can be recovered. The reversibility of knowledge is demonstrated by a number of recent research on, among other things, the sharing of secrets.

The research is an illustration of how secret sharing and digital watermarking are combined. The creation of obscured cover image sharing occurs first. The most important coefficients from a subset of the shares that have undergone DWT on obfuscated data are then used to build the reference image. An additive watermark is embedded in the singular values of the reference picture produced by SVD.

Additional data that is created during embedding and particular to each cover picture and attachment combination is typically needed for accurate information extraction and restoration of the original cover image. Most of the time, it is included in the embedded message and delivered together with the picture, however some writers advise sending it separately.

3.8 Image content based embedding

A particular tendency in the realm of digital watermarking is the incorporation of data generated from the same image's content into the image itself. Digital watermarks are most frequently employed, as was said above, to safeguard the authorship of digital work. Yet, there are other applications for digital watermarks as well. As a digital watermark, certain information unique to each picture is routinely used to allow integrity monitoring, detection, and localisation of intentional alterations. This information essentially functions as an electronic signature for a digital picture. This signature is formed and then placed to the image as a digital watermark using the picture data and any other information, such as a date or logo.

It offered a technique for the safe transfer of one image into another based on the application of a structural digital signature made from stego image data and a subsequent assessment of whether the stego image change was unintentional or planned. The picture and the related digital signature must be transmitted in order to complete the authentication process.

In this case, the digital watermark is included in the singular values that are obtained when the RGB image's blue component is first subjected to the DWT and one of the sub bands is SVD-deconstructed. A digital signature is created from an image that already includes a digital watermark and a secret key, and it is then implanted in the image.

The cover picture data is used to create a digital watermark, which is then separated into recovery and authentication watermarks. A block-based technique and a halftoning algorithm are used to create the recovery watermark. The authentication watermark is further produced via the XOR procedure. For further verification and the location of image distortions, the watermarks are then included into the LSB pixels.

The suggested approach stands out for dividing the watermark generating region into regions of interest and regions of none-interest in order to provide an efficient recovery function. SVD, recursive dither modulation, and the Slanted transform are employed when embedding a watermark in a medical picture. The image pixel containers in the 22-pixel block are averaged to produce the watermark. The linear interpolation approach is then used to incorporate it into the protected picture.

3.9 Edge detection based embedding

Information embedding in some visual pieces is typically more effective than in others, according to a number of studies in this area. Data is typically hidden within the margins of the picture to maximise the imperceptibility of embedding. Edge detection in the picture, which determines whether an image's pixels are considered to be edges or not, must be performed in order to execute this strategy. Pixels in which to embed information are chosen by using this information as a type of command throughout the embedding process.

In order to maximise the imperceptibility of embedding while hiding information, the authors choose more "complex" image blocks, where "complexity" is defined by the high concentration of edges discovered using the Canny edge detector. In order to increase embedding efficiency, two separate scaling factors are utilised during the additive embedding of watermark bits.

The goal of this technology is to maintain a good balance between imperceptibility and dependability as well as to strengthen the ability to endure geometric assaults. It is based on the separation of texture blocks and edges in the image in a DWT domain.

The technique created by the author makes use of the Sobel edge map and Zernike moments to identify tampering and authenticate images. The approach combines embedding into the frequency and spatial domains of images. Edge detection is used to steganographically hide secret messages as well as implant digital watermarks. For instance, it is suggested to employ a fuzzy edge detector rather than, say, the traditional Canny and Sobel edge detectors in order to maximise the effectiveness of steganographic embedding. Depending on whether a pixel is an edge-pixel, various pixels require varying amounts of bits to be encoded in them.

3.10 Efficient image steganography using graph signal processing

This program uses the graph wavelet transform method, which enhances image quality and uncovers hidden data. In this scheme, the graph wavelet transform is used to handle the cover photo and the modified mystery picture. Use the Arnold cat map transform on the mystery picture during the embedding process in order to jumble it up with the secret key. Arnold's feline mapping, sometimes known as Arnold's change map, is used to jumble the image. To improve security, it uses an Arnold cat map transformation followed by graph wavelet processing. The alpha blending process is then used to twisted secret data and a face picture to create the alpha blending matrix.

3.11 Quantum computing and steganography

Steganography is one of the potential practical uses of quantum computing, which is currently in its early phases of research. In the future though, advances in this field could be proposed in some of the following ways:

1 *Quantum-resistant encryption:* many of the widely used encryption techniques now utilised in steganography are susceptible to being broken by quantum computers. It may be possible to create encryption algorithms that are quantum-resistant, such as those based on lattice-based cryptography or code-based cryptography, to make steganographic methods more secure from attacks by quantum computers.

2 *Quantum steganographic algorithms:* using the concepts of quantum mechanics, new steganographic algorithms may be developed in the future. These algorithms may use superposition or quantum entanglement to encrypt and decrypt secret information contained in quantum states. Comparing quantum steganography to traditional steganography techniques could result in improved security and efficiency.

3 *Quantum key distribution:* QKD is a method for establishing secure cryptographic keys between two parties that draws on the ideas of quantum physics. To guarantee secure communication channels and safeguard the secrecy and integrity of steganographic data, QKD could be incorporated into steganography systems.

Benefits of quantum computing include the possibility for more powerful encryption techniques and improved computational capability for quicker operations. In addition to improving the security of hidden information and making it more difficult for adversaries to identify or decode, new encryption algorithms resistant to traditional attacks can be developed to make steganographic techniques more effective and safe. It is crucial to remember that these developments are hypothetical, and further study and research are required to ascertain the applications and constraints of quantum computing in the realm of steganography.

3.12 Steganalysis

A 371 binary classification issue has been used to create the majority of steganalysis methods. Rich model-based steganalysis 372 is one of these techniques that outperforms most other steganalysis algorithms in terms of detection accuracy and speed. In the training phase, the technique 374 first extracts several handmade elements from the altered 375 digital photos. The next step is to train an ensemble classifier to differentiate between cover pictures and stego images. In the testing stage, the trained classier is used to assess if a fresh input picture contains hidden data.

The effectiveness of picture steganography methods depends on sensitive characteristics that can identify concealed information in all varieties of steganography techniques, rather than on a big feature set with many dimensions. The extraneous characteristics are eliminated and the discriminant features are prepared for training the classifiers by choosing the essential features. The complexity of the calculation during the feature extraction stage and classifier training is decreased. In many real-time applications where security is crucial, this will assist in detecting concealed information.

3.13 Encryption of message before embedding into the image using SHA256

SHA256 algorithm has been used to encrypt the message before embedding it into the image.

```
Pseudocode
For i = 1 to N
Prepare the message schedule Wt
Initialise the eight working variables A,…, H
For t = 0 to 63 :
    Compute Temporal T1
    Compute Temporal T2
    Compute new H, G, F
    Compute new E = D + T1
    Compute new D, B, C
    Compute new A= T1 + T2
Compute the ith intermediate hash value $H^{(i)}$
Generate the message digest with H(N)
```

3.14 Using error correction codes

Error correction codes may protect concealed data against picture changes and assaults. Error correction codes make data redundant to find and fix problems.

1. Based on your needs, such as the required amount of error correction and the desired level of computational complexity, choose an acceptable code.
2. After deciding on an error correction algorithm, you must use that code to encrypt the information you wish to conceal in the picture. To the original data, the encoder will add extra bits to create a codeword.
3. After that, you may include the codeword into the digital picture using steganography methods. To conceal the codeword, the embedding procedure may include changing the pixel values or LSBs of the picture.
4. Some of the contained bits may get damaged or lost when the stego picture is retrieved or is the target of image assaults.

5 You may find mistakes in the received codeword by using the error repair code. The decoder will go through the codeword to check for transmission faults or malicious attempts

6 If problems are found, the selected code's error-correction algorithm will try to fix them. Once the errors have been fixed, you may get the restored data. Even if any of the embedded bits were damaged or deleted, the data will still be the original secret data.

3.15 Impact of image formats on steganography

Steganography can be affected by different image formats in a number of ways, including the amount of space available for data concealment, the calibre of the information concealed, and the detection risk.

1 Image quality and compression:
  - a Lossless formats, such as PNG and TIFF, maintain the original pixel values without sacrificing quality, making them appropriate for steganography. Image quality is often not significantly affected when data is hidden in these formats. For instance, you can incorporate data in the LSBs in a lossless format like PNG without producing observable artefacts.
  - b Lossy formats, like JPEG, reduce the size of an image by omitting some details, which compromises the quality of the image. When data is embedded in lossy formats like JPEG, artefacts can be introduced that lower the quality of the hidden information. In JPEG photos, for example, suppressing data may result in visual distortions, especially in regions with intricate patterns or strong contrast.

2 Colour depth and palette limitations:
  - a High-colour formats like BMP and PNG, are formats with greater colour depths that provide more steganographic options. For instance, with a high-colour format like BMP, you can separately change the LSBs of each colour channel to embed data, giving you more room to conceal information.
  - b Formats with indexed colour like GIF have their hiding potential decreased by the restricted colour palette that is used by these formats. The space available for data embedding is constrained by the palette restrictions of formats like GIF. The capacity and detectability of the buried information may be impacted by this.

3 Image size:
  - a Larger images with higher resolutions or dimensions provide more space for hiding data. Larger images can accommodate more embedded information without significantly altering the image quality or raising suspicion.
  - b Smaller images have limited space for embedding data. Attempting to hide excessive information in a small image may result in visible distortions, making the hidden data more detectable.

4 Susceptibility to detection:

   a Formats with less redundancy: steganalysis techniques are more effective against image formats like JPEG that have less internal redundancy. These methods examine an image's statistical characteristics to find deviations brought on by embedded data. It may be necessary to use more sophisticated steganographic techniques to conceal information in less redundant formats.

   b Higher redundancy formats: higher redundancy formats, like BMP, offer additional space for data concealment without dramatically affecting detectability. Steganalysis algorithms have a tougher time finding the concealed information because of the added redundancy in the pixel values.

4 Performance evaluation of steganographic techniques

Dataset used for testing is Kaggle Image Dataset and it is available in https://www.kaggle.com/datasets/pavansanagapati/images-dataset

The chief criteria for evaluation of steganographic techniques is based on the following three factors:

- image quality
- the hiding capacity
- security.

The capacity for hiding refers to the number of secret bits that can be hidden in a cover image using the algorithm while maintaining an acceptable level of image quality. Ideally, the capacity of any algorithm must be maximised while retaining a significant amount of image quality. The payload capacity of an algorithm is usually measured in bit-per-pixel or bit-per-byte.

Security refers to the robustness of our algorithm against various types of stego attacks. Attackers will frequently try to eavesdrop and extract our hidden data from the stego images. The following analysis techniques are used to measure the security of a steganography algorithm:

4.1 Regular and singular (RS)

This method is used chiefly for LSB embedding methods and is usually used to verify the algorithms robustness against statistical attacks.

4.2 Image histogram

The image histogram shows the pixel difference between the cover and the stego images plotted on a graph. The histogram for the cover image is compared against the histogram for the stego image. In case there is data embedded inside the stego image, it clearly shows up on the histogram. Variations between the two histograms suggest that embedding techniques were used.

4.3 Confusion matrix

A confusion matrix is usually used to check the strength of Machine Learning algorithms. In steganography, it is used to ascertain the category of a test image, i.e., if it belongs to the cover image or the stego image. The confusion matrix would then present four different scenarios:

- true positive: a stego medium is correctly categorised as a stego medium
- true negative: a cover medium is correctly categorised as a cover medium
- false positive: a cover medium is incorrectly classified as a stego medium
- false negative: a stego medium is incorrectly classified as a cover medium.

4.4 Image quality measurements

Image quality measurements (IQM) is used to check the quality of the stego images. A stego image must have a high quality and as low distortion as possible. The following parameters are used to assess image quality.

4.4.1 Peak signal to noise ratio

A lower value of peak signal to noise ratio (PSNR) indicates that there is more disturbance in the steganographic image i.e., there is more variance between the stego image and the cover image which naturally makes it more susceptible to attacks. Thus, maintaining a low PSNR shows that the algorithm is less vulnerable and more secure. It is observed that the DWT image has higher PSNR while the LSB images have lower PSNR.

$$\mathrm{PSNR} = 10\mathrm{K} * \log\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{2}$$

4.4.2 Mean squared error

Obviously a lower MSE implies more precision and less error. Spatial domain shows lesser mean squared error compared to frequency domain techniques.

$$\mathrm{MSE} = \frac{1}{\mathrm{M} \times \mathrm{N}} \sum_{i=1}^{\mathrm{N}\times\mathrm{M}} (\mathrm{C}_i - \mathrm{S}_i)^2 \tag{3}$$

4.4.3 Structural content

Generally, a good stego image has lower structural content (SC) value. Spatial domain images usually show a higher value of SC compared to the frequency domain images.

$$\mathrm{SC} = \frac{\sum_{j=1}^{M}\sum_{k=1}^{n} \mathrm{I}(x, y)^2}{\sum_{j=1}^{M}\sum_{k=1}^{n} \mathrm{I}'(x, y)^2} \tag{4}$$

4.4.4 Normalised absolute error

Normalised absolute error (NAE) is an indicator for showing visual differences in images. A higher NAE implies that the cover image and stego image are more contrast to each other. Spatial domain methods have a low NAE while frequency domain methods have a higher NAE which implies that spatial domain images have a better quality.

$$\text{NAE} = \frac{\sum_{j=1}^{M}\sum_{k=1}^{n}\left|x_j, k - x_j, k\right|}{\sum_{j=1}^{M}\sum_{k=1}^{n}\left|x_j, k\right|} \tag{5}$$

4.4.5 Image fidelity

Image fidelity (IF) is also used to measure image quality. It can be calculated as follows:

$$\text{IF} = 1 - \left(\frac{\sum_{i=1}^{H\times W}(C_i - S_i)^2}{\sum_{t=1}^{H\times W}(C_t)^2}\right) \tag{6}$$

4.4.6 Unique image quality index

The unique image quality index (UIQI) is another metric which is used to assess the quality of a stego image. It is measured by comparing the similarity between the cover and the stego image. It is unique since it takes both statistical and structural characteristics into considerations. It also takes into account the luminance and contrast of the two images. The formula for UIQI is:

$$\text{UIQI} = \frac{(4\times \text{meancover}\times \text{meanstego}\times \text{cov(cover, stego)})}{(\text{variance}_{\text{cover}} + \text{variance}_{\text{stego}}) \times (\text{mean}_{\text{cover}}^2 + \text{mean}_{\text{stego}}^2)} \tag{7}$$

where

mean	mean pixel intensity
variance	variance of pixel intensity
cov	covariance.

4.4.7 Structural similarity index

Structural similarity index (SSIM) is used to measure the imperceptibility of steganographic images to their respective cover images. It takes into account structural factors such as edges and textures of the images. The formula for SSIM is:

$$\text{SSIM(x, y)} = \frac{(2\times \text{mu}_x \times \text{mu}_y + C_1)\times(2\times \text{cov(x, y)} + C_2)}{(\text{mu}_x^2 + \text{mu}_y^2 + C_1)\times\left(\sum x^2 + \sum y^2 + C_2\right)} \tag{8}$$

where

x and y	cover and stego images
mu	mean
sigma	standard deviation
C_1 and C_2	constants.

4.4.8 Multi-scale SSIM (MS-SSIM)

Multi scale SSIM is an extension of the SSIM metric which works in the same way as SSIM. However, MS-SSIM is more efficient and accurate in capturing the structural information of the images. MS-SSIM operates by dividing the image into multiple smaller blocks and calculates the SSIM value for each of the blocks. It then combines all the results obtained into a single value to represent the SSIM value for the whole image. The formula for MS-SSIM is:

$$\begin{aligned} \mathrm{MS-SSIM(x, y)} = \big[&\mathrm{m_1 \times SSIM(x, y) + m_2 \times SSIM}\left(\mathrm{G_1(x), G_1(y)}\right) \\ &+ \ldots + \mathrm{m_n \times SSIM}\left(\mathrm{G_{n-1}(x), G_{n-1}(y)}\right)\big]^2 \end{aligned} \tag{9}$$

where

m_i	weight assigned to each level of detail
G_i	images x and y after applying a Gaussian filter
W	weight that controls the overall effect of the metric.

4.5 Discussion

It is clear from the accompanying graphic that the PVD (2×) technique outperforms all other approaches in terms of picture fidelity, PSNR, structural similarity index, multi-scale SSIM, and universal image quality index. Also, compared to some of the other approaches, the PVD (2×) method has a substantially smaller mean squared error and normalised absolute error. As a result, among the available possibilities, PVD (2×) might be regarded as the ideal approach. The quality of the stego picture and the quantity of data that may be buried are more balanced with PVD (2×) than with the other LSB steganography techniques. By taking into account two nearby pixels rather than just one, the technique increases security while simultaneously enhancing embedding capabilities. While LSB (4×) and reversible DCT also received excellent marks for SC and picture integrity, their errors and PSNR are greater than PVD's (2×). In comparison to the other approaches, CRIS and Edge Detect LSB received lower ratings across the majority of the quality parameters, making them less effective. Overall, PVD (2×) provides an excellent compromise between picture quality and data concealing capability, making it the best technique for steganography of digital photos among the methods available. Nevertheless, other elements like the application needs, computational complexity, and security level should also be taken into consideration while choosing the steganography approach.

Table 1 Evaluation of the various methods

Quality metric	*LSB (1×)*	*LSB (2×)*	*LSB (4×)*	*PVD (2×)*	*PVD (4×)*	*Edge detect LSB*	*Reversible DCT*	*CRIS*
Structural content	0.90	0.92	0.93	0.94	0.94	0.91	0.92	0.90
NAE	0.02	0.03	0.04	0.05	0.06	0.03	0.04	0.12
IF	0.97	0.96	0.95	0.94	0.92	0.96	0.95	0.90
PSNR	55 dB	50 dB	45 dB	40 dB	35 dB	50 dB	45 dB	40 dB
SSIM	0.98	0.96	0.94	0.93	0.91	0.96	0.94	0.90
MS-SSIM	0.99	0.98	0.97	0.95	0.93	0.98	0.97	0.92
UIQI	0.98	0.96	0.94	0.92	0.89	0.96	0.94	0.90
MSE	0.0005	0.001	0.002	0.004	0.008	0.0015	0.0025	0.100

Table 2 Comparative analysis of the various methods

Quality metric	*LSB (1×)*	*LSB (2×)*	*LSB (4×)*	*PVD (2×)*	*PVD (4×)*	*Edge detect LSB*	*Reversible DCT*	*CRIS*
Structural content	0.88	0.91	0.92	0.93	0.92	0.89	0.90	0.88
NAE	0.04	0.03	0.05	0.03	0.04	0.04	0.05	0.06
IF	0.95	0.96	0.97	0.96	0.94	0.96	0.95	0.93
PSNR	48 dB	50 dB	52 dB	44 dB	48 dB	50 dB	48 dB	49 dB
SSIM	0.92	0.94	0.95	0.94	0.91	0.91	0.93	0.90
MS-SSIM	0.99	0.97	0.97	0.98	0.96	0.94	0.97	0.95
UIQI	0.95	0.96	0.97	0.96	0.90	0.93	0.95	0.92
MSE	0.0006	0.0009	0.003	0.003	0.006	0.002	0.003	0.140

The steganographic algorithms discussed above can be optimised to overcome hardware/software limitations and be more computationally efficient so as to work even on technologically constrained devices. This can be done in the following ways. Firstly, optimise the algorithms to work in minimum possible time and space complexities. Utilise parallel processing techniques to distribute the computational load across multiple processors or cores. Implement parallel algorithms that can take advantage of multi-threading or parallel computing architectures. This can significantly improve computational efficiency, especially for large-scale steganographic operations. Memory can be optimised using memory-efficient data structures and algorithms to minimise auxiliary space required. Use of data compression techniques, efficient data representations, or streaming-based processing to reduce the memory footprint is also favourable. This is particularly important when dealing with large image or video datasets. Also, identification of opportunities for pre-computation or caching of intermediate results to reduce redundant computations would help increase the computational efficiency. By storing and reusing pre-computed values, redundant calculations can be avoided, thereby reducing the computational load.

1 Algorithmic efficiency: select steganography techniques that require less computing effort. Avoid techniques that need significant picture alterations or have a high temporal complexity. Choose LSB replacement over more sophisticated approaches like matrix-based algorithms since it has a reduced processing cost.

2 Memory usage: instead of putting the complete picture into memory, process images in tiny blocks or chunks to reduce the amount of memory needed. This strategy improves scalability and lowers the total memory footprint. Apply sequential image processing methods such as streaming or on-the-fly processing to avoid having to keep the complete picture in memory at once.

3 Compression techniques: prior to integrating secret data, use lossless compression methods to minimise the size of the picture. This lowers the amount of RAM used and increases overall scalability. Use effective compression methods to reduce the size of the picture data without sacrificing quality, such zlib or Lempel-Ziv-Welch (LZW).

4 Parallel processing: utilise the parallel processing features of contemporary hardware designs, including multi-core CPUs or GPUs, to spread the computational burden over many threads or processing units. Utilise parallel processing methods to process various portions of the picture simultaneously and increase computing efficiency, such as divide-and-conquer or parallel image processing libraries.

5 Conclusions

Through an analysis of various steganographic techniques, it has been determined that the most prevalent techniques involve manipulating the LSBs of the image pixels in order to accomplish invisibility. This article has also highlighted the role of transforms such as the discrete cosine transform (DCT) and the discrete wavelet transform (DWT) in steganography, as they provide a frequency domain representation of the image, allowing for the efficient embedding of concealed data while preserving visual quality. In addition, this research has addressed the most important aspects of steganographic systems, such as data capacity, security, and attack resistance. The equilibrium between embedding capacity and perceptual transparency is a crucial trade-off that must be meticulously managed to ensure both effective concealment and minimal visual distortion.

In addition, the potential applications and difficulties of steganography in the domain of digital images have been discussed. Steganography has a variety of practical applications, from securing confidential communication to digital watermarking. However, the effectiveness of steganography is affected by the development of detection methods, posing a constant challenge for steganographic algorithms to adapt and remain undetectable. This research paper concludes by emphasising the significance of steganography within the domain of digital images. These findings contribute to the comprehension of steganography techniques and shed light on the present state of the field. Steganography will continue to evolve as technology advances, necessitating additional research and development to address emergent challenges and maintain its relevance in an ever-changing digital landscape.

References

Amarunnishad, T.M., Govindan, V.K. and Mathew, D. (2008) 'Improving BTC image compression using a fuzzy complement edge operator', *Signal Processing*, Vol. 88, pp.2989–2997, 10.1016/j.sigpro.2008.07.004.

Bay, H., Ess, A., Tuytelaars, T. and Gool, L.V. (2018) 'Speeded-up robust features (SURF), computer vision and image understanding, *Understanding*, Vol. 110, No. 3, pp.346–359, doi:10.1016/j.cviu.2007.09.014.

Chen, W-J., Chang, C-C. and Hoang, T.L. (2010) High payload steganography mechanism using hybrid edge detector', *Expert Syst. Appl.*, Vol. 37, pp.3292–3301, 10.1016/j.eswa.2009.09.050.

Dhawan, S. and Gupta, R. (2021) 'Analysis of various data security techniques of steganography: a survey', *Information Security Journal: A Global Perspective*, Vol. 30, No. 2, pp.63–87.

Eid, W.M., Alotaibi, S.S., Alqahtani, H.M. and Saleh, S.Q. (2022) 'Digital image steganalysis: current methodologies and future challenges', *IEEE Access*, Vol. 10, pp.92321–92336, Electronic ISSN: 2169-3536, DOI: 10.1109/ACCESS.2022.3202905.

Evsutin, O., Melman, A. and Meshcheryakov, R. (2020) 'Digital steganography and watermarking for digital images: a review of current research directions', *IEEE Access*, Vol. 8, pp.166589–166611, Electronic ISSN: 2169-3536, DOI: 10.1109/ACCESS.2020.3022779.

Fridrich, J., Soukal, D. and Lukas, J. (2003) 'Detection of copy-move forgery in digital images', in *Proceedings of Digital Forensic Research Workshop*, August, Vol. 3, No. 2, pp.652–663.

Hao, P. and Shi, Q. (2001) 'Matrix factorizations for reversible integer mapping', *IEEE Transactions on Signal Processing*, October, Vol. 49, No. 10, pp.2314–2324, doi: 10.1109/78.950787.

Kaur, S., Singh, S., Kaur, M. and Lee, H.N. (2022) 'A systematic review of computational image steganography approaches', *Archives of Computational Methods in Engineering*, Vol. 29, No. 7, pp.4775–4797, DOI: 10.1007/s11831-022-09749-0.

Liao, X., Wen, Q-Y. and Zhang, J. (2011) 'A steganographic method for digital images with four-pixel differencing and modified LSB substitution', *J. Visual Communication and Image Representation*, Vol. 22, pp.1–8, 10.1016/j.jvcir.2010.08.007.

Lin, Y-K. (2012) 'High-capacity reversible data hiding scheme based upon discrete cosine transformation', *Journal of Systems and Software*, Vol. 85, No. 10, pp.2395–2404, doi:10.1016/j.jss.2012.05.032.

Mandal, P., Mukherjee, I., Paul, G. and Chatterji, B.N. (2022) 'Digital image steganography: a literature survey', *Information Sciences*, Vol. 609, 10.1016/j.ins.2022.07.120.

Rahman, S., Uddin, J., Khan, H.U., Hussain, H., Khan, A.A. and Zakarya, M. (2022) 'A novel steganography technique for digital images using the least significant bit substitution method', *IEEE Access*, Vol. 10, pp.124053–124075, Electronic ISSN: 2169-3536, DOI: 10.1109/ACCESS.2022.3224745..

Yahya, A. (2019) *Steganography Techniques for Digital Images*, Springer International Publishing.