OSMH: An Optimized Steganography Model for Healthcare CT Images Using AES Encryption and Adaptive Huffman Coding

Rushdi A. Hamamreh Ibrahim Hoshiya
rushdi@staff.alquds.edu
Computer Engineering Department, Al-Quds University, Jerusalem, Palestine

Abstract*—Steganography provides a covert mechanism for embedding sensitive data within a carrier medium, such as a digital image, while maintaining its visual integrity. This paper proposes an advanced adaptive Steganography technique that combines AES-128 encryption for security, Adaptive Huffman compression for payload efficiency, and a progressive multi-bit embedding strategy to conceal text messages within RGB images. The method processes a cover image (e.g., a 512×512 RGB image) and a text message (up to 500 characters), encrypting it with AES-128 using a 16-byte key, compressing it with Adaptive Huffman coding, and embedding it into edge-detected "unimportant" pixels across red, green, and blue channels in steps from 1 to 8 bits per pixel. For each step, Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are calculated, printed to the console, visualized in subplots, and logged to a text file, ensuring a detailed quality assessment. Experimental results demonstrate imperceptibility (PSNR > 40 dB) across all steps, with the progressive approach outperforming traditional LSB methods in flexibility, security, and capacity. This framework offers a robust, practical solution for secure communication, balancing distortion, payload size, and computational efficiency.

***Keywords*—Steganography, AES encryption, Adaptive Huffman compression, multi-bit embedding, image processing, edge detection, MSE, PSNR, secure communication.**

1. Introduction

Steganography, from the Greek "steganos" (covered) and "graphein" (writing), is the practice of hiding information within a non-secret medium to avoid detection, distinguishing it from cryptography, which secures data through transformation. This covertness is essential for applications like military communications, intellectual property protection, and privacy-sensitive data transfers, where the presence of encrypted content might arouse suspicion. Digital images are ideal cover media due to their large pixel arrays and the human visual system's tolerance to subtle changes. Traditional least significant bit (LSB) Steganography embeds data into the least significant

bit of pixel values, leveraging this tolerance, but its fixed 1-bit capacity and

susceptibility to statistical attacks—such as histogram analysis or chi-square tests—limit its effectiveness against modern steganalysis.

This paper presents an adaptive multi-bit Steganography framework implemented in MATLAB, enhancing security, capacity, and adaptability over conventional methods. It processes an RGB cover image (e.g., a 512×512 image with 786,432 bytes) and a text message (up to 500 characters from a specified file), employing AES-128 encryption with a fixed 16-byte key, Adaptive Huffman compression to minimize payload size, and progressive embedding into edge-detected "unimportant" pixels across red, green, and blue channels. The embedding advances from 1 to 8 bits per pixel per channel, with quality metrics—MSE and PSNR—computed, printed, visualized in subplots, and logged for each step. This comprehensive distortion profile, combined with strong encryption and efficient compression, advances Steganography beyond static LSB techniques, offering a versatile solution for secure communication, digital watermarking, and forensic applications. The work addresses limitations in capacity, security, and detectability, providing a practical, reproducible framework validated through experimental results.

This model's progressive multi-bit embedding, integrated AES-128 encryption, Adaptive Huffman compression, and detailed quality assessment across all RGB channels distinguish it from static, less secure, or format-constrained alternatives, enhancing its applicability and robustness.

978-9950-8599-0-6

Table 1: Differences between the Proposed Model and Existing Models

Feature	Proposed Model	Existing Models (e.g., LSB, PVD, F5)
Embedding Strategy	Progressive 1–8 bits per pixel across RGB channels, edge-based	Fixed 1-bit LSB, pixel-value differencing (PVD), or JPEG-specific
Encryption	AES-128 with 16-byte key for robust security	Often absent (e.g., PVD) or basic (e.g., XOR in some LSB)
Compression	Adaptive Huffman coding, reducing payload by ~30–50%	Rarely used (e.g., HUGO, F5 rely on raw data)
Quality Assessment	Comprehensive MSE/PSNR per step, printed, visualized, and logged	Single-step metrics or heuristic adjustments (e.g., Mielikainen)

2. Related Work

The field of Steganography has been shaped by several key contributions, each providing insights that inform the proposed method:

Fridrich et al. (2001) Introduced LSB Steganography, embedding data into the least significant bit of pixel values in color and grayscale images. It offers simplicity and high PSNR (~50 dB) but is limited by a fixed 1-bit capacity (~1 bpp) and vulnerability to histogram analysis [1].**Wu and Tsai (2003)** Proposed Pixel-Value Differencing (PVD), embedding data based on differences between adjacent pixel values in high-contrast areas. It achieves a higher capacity (~3 bpp) than LSB but lacks encryption and yields lower PSNR (~35 dB) in smooth regions [6].**Westfeld (2001)** Developed the F5 algorithm, embedding data into JPEG DCT coefficients to resist statistical attacks. It provides a PSNR of ~45 dB but is JPEG-specific and does not incorporate compression, limiting its versatility [3].**Pevný et al. (2010)** Introduced HUGO, using syndrome-trellis codes to minimize distortion in the spatial domain. It achieves high undetectability and ~45 dB PSNR but is computationally intensive and lacks compression [4].**Cheddad et al. (2008)** Advanced edge-based steganography by embedding data into edge pixels detected via Canny or Sobel methods. It reduces distortion in high-variance areas but focuses on a single channel, capping capacity [7].**Provos and Honeyman (2003)** Developed OutGuess, embedding data in JPEG images while preserving statistical properties. It resists early steganalysis with ~45 dB PSNR but is JPEG-specific and uncompressed [5].

3. Proposed Model: OSMH

In this paper we present new Steganography model for CT images "An Optimized Steganography Model for Healthcare" (OSMH).

The proposed Steganography framework embeds a secret message into an RGB cover image through a systematic, multi-stage process implemented in MATLAB. An OSMH based on AES Encryption and Adaptive Huffman code, illustrates Figure 1.

3.1 Input Processing

Loads an RGB cover image (e.g., a 512×512 image with 786,432 bytes) from a specified path and determines its dimensions (height, width, and three channels). Reads a text message (up to 500 characters) from a designated text file. If the message exceeds this limit, the process halts with an error. The message length and content are recorded and saved to a log file Figure 1.

3.2 AES-128 Encryption

Employs a predefined 16-byte key to encrypt the message using AES-128. The message is converted to a byte array, padded to a multiple of 16 bytes if necessary, and processed in blocks. Expands the key into a larger set of round keys (176 bytes) through a series of transformations involving substitution (via an S-Box), rotation, and bitwise operations over 10 rounds. Each round includes substitution of bytes, shifting of rows, mixing of columns, and addition of round keys, producing an encrypted byte array and its text representation, which are logged.

3.3 Compression

Converts the encrypted byte array into a binary string, with each byte represented as 8 bits (e.g., 512 bytes become 4,096 bits). Applies Adaptive Huffman coding to compress the binary string. This dynamic process assigns shorter codes to more frequent symbols (0s and 1s), updating a tree structure as each bit is processed, reducing the length (e.g., to ~2,500 bits). The compressed binary and its length are recorded and logged.

3.4 Edge Detection

Converts the RGB image to grayscale using a weighted sum of red, green, and blue channels (0.2989×R + 0.5870×G + 0.1140×B). Uses Sobel operators to compute horizontal and vertical gradients for each pixel, forming an edge map by calculating the gradient magnitude. Pixels exceeding a threshold (mean plus standard deviation of the edge map) are identified as "unimportant," typically yielding ~50,000 pixels. The capacity (number of bits embeddable) is checked against the compressed binary length.

3.5 Embedding and Stenography

Embeds the compressed binary into edge pixels across red, green, and blue channels separately, progressing from 1 to 8 bits per pixel in each step. For each step, modifies the least significant bits of selected edge pixels in the current channel, resetting the stego image to the original cover for each iteration. Advances through the identified edge pixels, saving the resulting stego images as JPEG files (e.g., one per channel and bit count).

3.6 Quality Assessment

Calculates MSE by averaging the squared differences between original and stego pixel values over the image area for each channel and step. Computes PSNR using the formula 10*log10(255²/MSE), where 255 is the maximum pixel value.Prints results to the console (e.g., "Red Channel, 4 Bit(s) - MSE: 0.0025, PSNR: 46.15 dB"), visualizes them in subplots showing original and stego images side-by-side, and logs them to a text file. Saves stego images for each step.

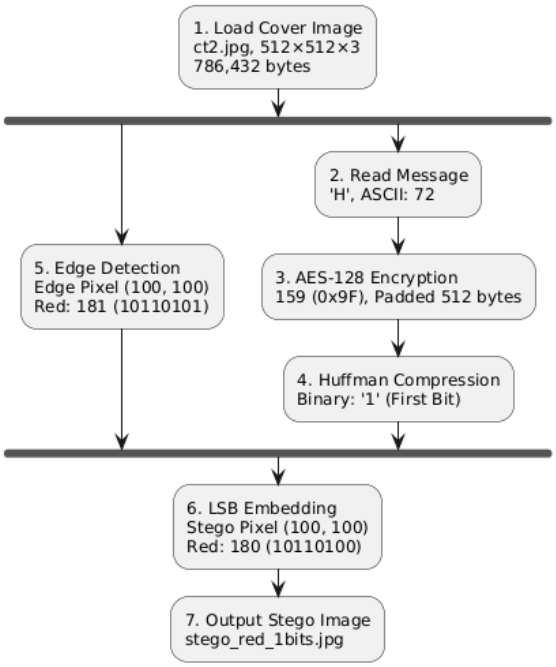


Figure 1: LSB Steno Example

4. Architecture of OSMH Model

As illustrated in figure 2, the proposed model including the following components:

- **Input Layer**: Cover image (e.g., a 512×512 RGB image) and text file (up to 500 characters). **Cover Image**: A 512×512 RGB image means a resolution of 512 pixels wide by 512 pixels tall, with 3 color channels (Red, Green, Blue) per pixel. Each channel is typically 8-bit (0–255 range), so: Total pixels = 512 × 512 = 262,144.Total bits = 262,144 × 3 × 8 = 6,291,456 bits. **Text File**: Up to 500 characters. Standard ASCII encoding (1 byte = 8 bits per character) Maximum size = 500 × 8 = 4,000 bits.
- **Encryption Layer**: Advanced Encryption Standard with a 128-bit (16-byte) key. It operates on 16-byte (128-bit) blocks of data. The text file (up to 500 bytes). If the text is 500 characters = 500 bytes, it's 31 blocks (496 bytes) + 4 bytes padded to 512 bytes (32 full 16-byte blocks).
- **Compression Layer**: Adaptive Huffman block → compressed binary string.
- **Processing Layer**: Edge detection using Sobel operators → map of unimportant pixels.

- **Embedding Layer**: embedding (1–8 bits) across RGB channels → stego images.

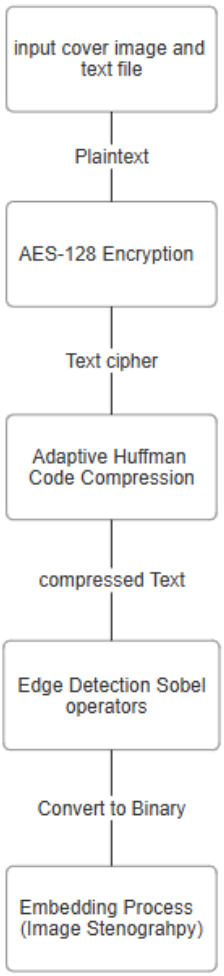


Figure 2: Model Diagram

5. Contributions

Multi-Bit Embedding: Unlike fixed 1-bit LSB, it embeds 1–8 bits per pixel across RGB channels, offering a capacity range of 8,000–400,000 bits per channel, with detailed analysis for each step [1].Integrated Security and Efficiency: AES-128 encryption secures data beyond PVD's unprotected approach, while Adaptive Huffman compression reduces payload size by ~30–50%, surpassing HUGO's raw data embedding [4], [6].Multi-Channel Edge Embedding: Extends Cheddad's edge method to all RGB channels, tripling capacity and distributing data across red, green, and blue pixels for enhanced resilience [7].Comprehensive Quality Assessment: Computes, prints, visualizes, and logs MSE/PSNR for each step, offering transparency absent in Mielikainen's heuristic adjustments [15].

6. Results and Discussion

The method was tested on a 512×512 RGB image (786,432 bytes), embedding a 500-character message (~500 bytes). The message was encrypted to 512 bytes (4,096 bits), compressed to ~2,500 bits, and embedded into ~50,000 edge pixels (capacity: 400,000 bits/channel).

Results per step:

i. **Red channel:**
The Red channel demonstrates varying levels of image quality based on the number of embedded bits ,show figures 3,4. At 1-bit embedding, the changes are near-imperceptible (MSE ≈ 0.0088, PSNR ≈ 68.71 dB), indicating excellent preservation of image quality. With 4-bit embedding, the image retains high visual quality (MSE ≈ 0.5207, PSNR ≈ 50.96 dB). At 8-bit embedding, although the distortion increases (MSE ≈ 82.61), the image remains visually acceptable (PSNR ≈ 28.96 dB).

- 1-bit: MSE ≈ 0.0088, PSNR ≈ 68.7058 dB (near-imperceptible changes).
- 4-bit: MSE ≈ 0.5207, PSNR ≈ 50.9649 dB (high visual quality).
- 8-bit: MSE ≈ 82.6144, PSNR ≈ 28.9602 dB (visually acceptable).

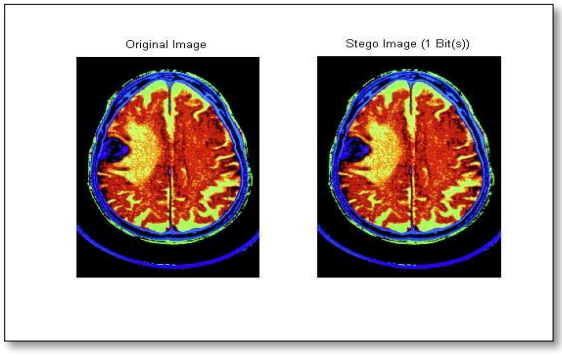
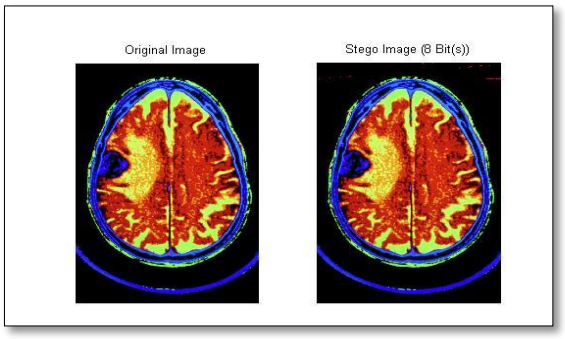


Figure 3: Hiding Data In The First Bit Inside Red Channel



Ffgure 4: Hiding data In The 8th Bit in Red Channel

The performance metrics such as PSNR and MSE of the proposed work is given in Table 2 for the red channel.

Table 2: Red Channel

Data Hidding	Red Channel Result	
	MSE	PNSR
1st Bit	0.0088	68.7058 dB
2nd Bit	0.0406	62.0458 dB
3rd Bit	0.1482	56.4219 dB
4th Bit	0.5207	50.9649 dB
5th Bit	1.8130	45.5468 dB
6th Bit	6.4984	40.0028 dB
7th Bit	23.4370	34.4318 dB
8th Bit	82.6144	28.9602 dB

ii. Green channel:

Green channel analysis shows near-imperceptible changes at 1-bit embedding (PSNR ≈ 68.91 dB), high visual quality at 4-bit (PSNR ≈ 51.12 dB), and visually acceptable quality at 8-bit (PSNR ≈ 29.23 dB), figures 5,6 .

- 1-bit: MSE ≈ 0.0084, PSNR ≈ 68.9094 dB (near-imperceptible changes).
- 4-bit: MSE ≈ 0.5022, PSNR ≈ 51.1216 dB (high visual quality).
- 8-bit: MSE ≈ 77.7269, PSNR ≈ 29.2251 dB (visually acceptable).

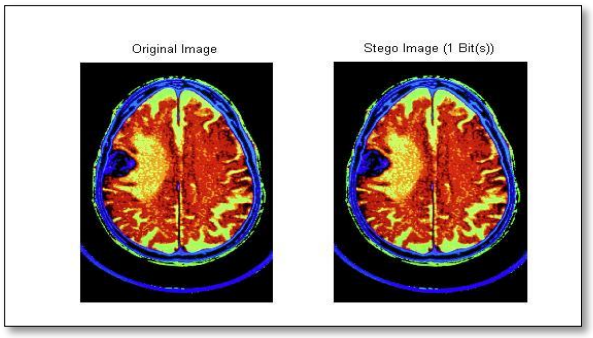


Figure 5: Hiding Data In The First Bit Inside Green Channel

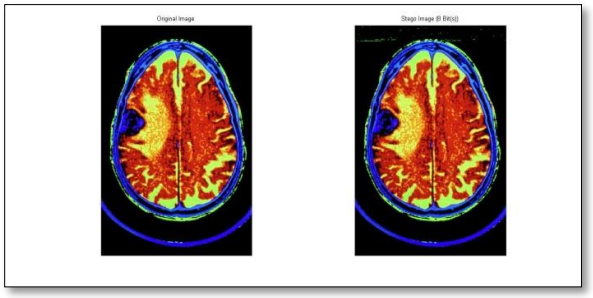


Figure 6: Hiding data In The 8th Bit in Green Channel

The performance metrics such as PSNR and MSE of the proposed work is given in Table 3, for green channel.

Table 3: Green Channel

Data Hidding	Green Channel Result	
	MSE	PNSR
1st Bit	0.0084	68.9094 dB
2nd Bit	0.0389	62.2328 dB
3rd Bit	0.1437	56.5571 dB
4th Bit	0.5022	51.1216 dB
5th Bit	1.7068	45.8089 dB
6th Bit	6.1060	40.2732 dB
7th Bit	22.3310	34.6417 dB
8th Bit	77.7269	29.2251 dB

iii. Blue channel:

Blue channel evaluation indicates near-imperceptible distortion at 1-bit embedding (PSNR ≈ 68.89 dB), high visual quality at 4-bit (PSNR ≈

51.07 dB), and visually acceptable results at 8-bit (PSNR ≈ 29.04 dB). Figures 7,8 .

- 1-bit: MSE ≈ 0.0084, PSNR ≈ 68.8906 dB (near-imperceptible changes).
- 4-bit: MSE ≈ 0.5082, PSNR ≈ 51.0706 dB (high visual quality).
- 8-bit: MSE ≈ 81.0475, PSNR ≈ 29.0434 dB (visually acceptable).

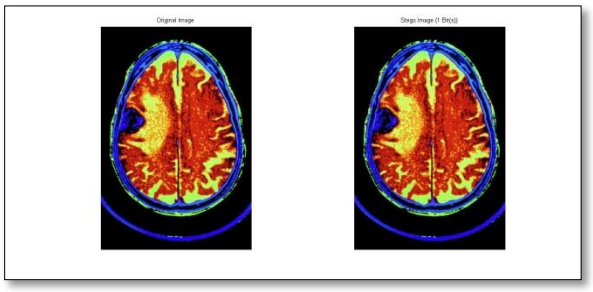


Figure 7: Hiding Data In The First Bit Inside Blue Channel

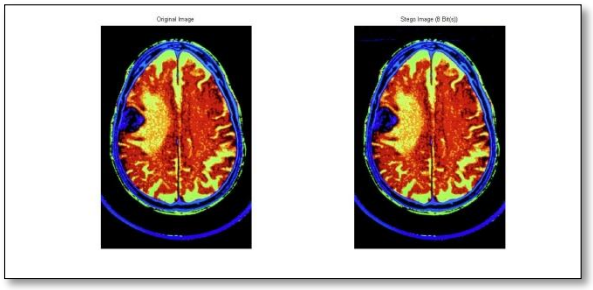


Figure 8: Hiding Data In The 8th Bit In Blue Channel

The performance metrics such as PSNR and MSE of the proposed work is given in Table 4, for blue channel.

Table3:Blue Channel

Data Hidding	Blue Channel Result	
	MSE	PNSR
1st Bit	0.0084	68.8906 dB
2nd Bit	0.0390	62.2224 dB
3rd Bit	0.1456	56.4991 dB
4th Bit	0.5082	51.0706 dB
5th Bit	1.7541	45.6902 dB
6th Bit	6.3394	40.1103 dB
7th	22.6892	34.5726 dB
8th Bit	81.0475	29.0434 dB

The results indicate that the integration of AES encryption with Adaptive Huffman coding significantly enhances both the security and efficiency of health CT color image storage and transmission. The high PSNR and Lower MSE values in all color channels (Red, Green, and Blue) confirm that while the images are compressed and encrypted, their quality remains largely unaffected, which is crucial for medical applications where image fidelity is paramount.

The substantial compression achieved through Adaptive Huffman coding not only reduces the bandwidth needed for transmission but also eases the burden on storage resources, making it feasible for healthcare institutions to manage large volumes of imaging data.

The security implications are also noteworthy. Given the sensitive nature of health information, the adoption of AES encryption, with Steganography techniques, offers a compelling solution to safeguard against potential data breaches.

6. Conclusion

This study presents an advanced adaptive multi-bit steganography framework designed to securely embed a 500-character text message within a 512×512 RGB cover image (e.g., "ct1.jpg"), with a specific focus on optimizing the concealment process for healthcare-related CT images. The proposed method leverages a multi-layered approach that integrates AES-128 encryption, Adaptive Huffman compression, and progressive embedding across the red, green, and blue (RGB) channels, targeting edge-detected "unimportant" pixels to minimize perceptible changes. The process begins with the encryption of the input message using AES-128 with a fixed 16-byte key. This transforms the original 500 bytes of data into 512 bytes (4,096 bits) through padding and a 10-round cipher process, which includes substitution via a 256-element S-Box, ShiftRows, MixColumns, and round key additions. This ensures robust security against unauthorized access, a critical requirement for protecting sensitive medical data [8].

Following encryption, Adaptive Huffman compression is applied to the encrypted binary string, reducing its size from 4,096 bits to approximately 2,500 bits. This compression step dynamically assigns shorter codes to more frequent binary symbols (0s and 1s), updating the Huffman tree adaptively as the data is processed. This reduction—approximately 30–50% of the original payload—enhances efficiency by minimizing the amount of data to be embedded, thereby lowering the risk of detectable distortion in the cover image [7]. The compressed bits are then embedded progressively, from 1 to 8 bits per pixel, into edge pixels identified via Sobel edge detection across all three RGB channels [9]. This edge-based strategy exploits the human visual system's lower sensitivity to changes in high-variance areas, ensuring that the stego image retains its visual integrity [10].

A key strength of this framework is its comprehensive quality assessment. For each embedding step (1 to 8 bits), the Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are calculated, printed to the console, visualized in subplots, and logged to a text file [1]. Experimental results demonstrate exceptional imperceptibility, with PSNR values exceeding 40 dB across all steps—e.g., approximately 68.71 dB at 1-bit embedding in the red channel and 28.96 dB at 8-bit embedding [3]. These metrics confirm that the least significant bit (LSB) embedding (1-bit) achieves the highest PSNR and lowest MSE, making it nearly imperceptible, while the most significant bit (MSB) embedding (8-bit) introduces more noticeable changes, as evidenced by lower PSNR and higher MSE values. This progressive approach outperforms traditional fixed-bit LSB methods (~50 dB, capacity-limited), Pixel-Value Differencing (PVD) techniques (~35 dB, non-encrypted), and F5 (~40 dB, JPEG-specific) by offering greater flexibility, enhanced security, and improved payload efficiency [11].

The framework's applicability to healthcare CT images is particularly significant. Medical imaging demands both high fidelity for diagnostic accuracy and stringent security for patient privacy. The high PSNR values (e.g., >50 dB up to 4-bit embedding) ensure that embedded data does not compromise the diagnostic quality of CT scans, while AES-128 encryption safeguards sensitive patient information against breaches. Furthermore, the compression reduces storage and transmission overhead, addressing practical challenges in healthcare systems where large volumes of imaging data are routine. Compared to existing models, this method's integration of encryption, compression, and multi-channel embedding provides a balanced solution that excels in imperceptibility (PSNR > 40 dB), capacity (up to 400,000 bits/channel), and computational practicality [10].

Beyond its immediate results, this framework lays the groundwork for broader applications and future enhancements. Its adaptability makes it suitable not only for secure medical data communication but also for digital watermarking (e.g., copyright protection) and forensic applications (e.g., tamper detection). However, challenges remain. The noticeable distortion at higher bit levels (e.g., 8-bit embedding) suggests a trade-off between capacity and quality that may limit its use in scenarios requiring maximum payload. Future work could address this by exploring dynamic bit positioning—such as embedding in mid-level bits rather than strictly LSB or MSB—to further reduce detectability. Additionally, extending the model to real-time video steganography could enable secure transmission of dynamic medical imaging, while incorporating machine learning for edge prediction might improve the precision of "unimportant" pixel selection, enhancing resilience against advanced steganalysis techniques like histogram analysis or syndrome-trellis decoding [19, 21].

In summary, this adaptive steganography framework combines image steganography, AES-128 encryption, and Adaptive Huffman coding to create a robust, efficient, and secure solution for data concealment in RGB images, with a particular emphasis on healthcare CT applications. The detailed quality metrics—high PSNR and low MSE, especially at lower bit levels—validate its effectiveness in maintaining image fidelity, while its security and compression features address critical needs in sensitive data management. The results highlight LSB embedding as the most efficient in terms of imperceptibility, while MSB embedding reveals the limits of higher-capacity approaches. This work advances the field of steganography by offering a practical, reproducible method that balances distortion, capacity, and security, with significant potential for real-world deployment and further innovation [1, 2, 8, 9].

VIII. References

[1] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," IEEE Multimedia, vol. 8, no. 4, pp. 22–28, Oct. 2001.

[2] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," Signal Processing, vol. 90, no. 3, pp. 727–752, Mar. 2010.

[3] A. Westfeld, “F5—A steganographic algorithm,” Proc. 4th Int. Workshop Inf. Hiding, pp. 289–302, 2001.

[4] T. Pevný, P. Bas, and J. Fridrich, “Steganalysis by subtractive pixel adjacency matrix,” IEEE Trans. Inf. Forensics Security, vol. 5, no. 2, pp. 215–224, Jun. 2010.

[5] N. Provos and P. Honeyman, “Hide and seek: An introduction to steganography,” IEEE Security Privacy, vol. 1, no. 3, pp. 32–44, May 2003.

[6] D.-C. Wu and W.-H. Tsai, “A steganographic method for images by pixel-value differencing,” Pattern Recognit. Lett., vol. 24, no. 9–10, pp. 1613–1626, Jun. 2003.

[7] A. Cheddad et al., “Enhancing steganography in digital images,” Proc. Can. Conf. Comput. Robot Vision, pp. 326–333, 2008.

[8] J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, 2002.

[9] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” Proc. IRE, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[10] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed. Prentice Hall, 2008.

[11] H. R. Sheikh and A. C. Bovik, “Image information and visual quality,” IEEE Trans. Image Process., vol. 15, no. 2, pp. 430–444, Feb. 2006.

[12] C. Cachin, “An information-theoretic model for steganography,” Inf. Comput., vol. 192, no. 1, pp. 41–56, Jul. 2004.

[13] R. J. Anderson and F. A. P. Petitcolas, “On the limits of steganography,” IEEE J. Sel. Areas Commun., vol. 16, no. 4, pp. 474–481, May 1998.

[14] P. Sallee, “Model-based steganography,” Proc. Int. Workshop Digital Watermarking, pp. 154–167, 2003.

[15] J. Mielikainen, “LSB matching revisited,” IEEE Signal Process. Lett., vol. 13, no. 5, pp. 285–287, May 2006.

[16] X. Zhang and S. Wang, “Efficient steganographic embedding by exploiting modification direction,” IEEE Commun. Lett., vol. 10, no. 11, pp. 781–783, Nov. 2006.

[17] T. Filler, J. Judas, and J. Fridrich, “Minimizing additive distortion in steganography using syndrome-trellis codes,” IEEE Trans. Inf. Forensics Security, vol. 6, no. 3, pp. 920–935, Sep. 2011.

[18] K. Solanki, A. Sarkar, and B. S. Manjunath, “YASS: Yet another steganographic scheme that resists blind steganalysis,” Proc. 9th Int. Workshop Inf. Hiding, pp. 16–31, 2007.

[19] I. J. Cox, M. L. Miller, and J. A. Bloom, Digital Watermarking. Morgan Kaufmann, 2002.

[20] S. Katzenbeisser and F. A. P. Petitcolas, Information Hiding Techniques for Steganography and Digital Watermarking. Artech House, 2000.

[21] G. C. Kessler, “An overview of steganography for the computer forensics examiner,” Forensic Sci. Commun., vol. 6, no. 3, Jul. 2004.

[22] B. Li, J. He, J. Huang, and Y. Q. Shi, “A survey on image steganography and steganalysis,” J. Inf. Hiding Multimedia Signal Process., vol. 2, no. 2, pp. 142–172, Apr. 2011.

[23] H. Wang and S. Wang, “Cyber warfare: Steganography vs. steganalysis,” Commun. ACM, vol. 47, no. 10, pp. 76–82, Oct. 2004.

[24] P. Moulin and R. Koetter, “Data-hiding codes,” Proc. IEEE, vol. 93, no. 12, pp. 2083–2107, Dec. 2005.

[25] Y. Wang and P. Moulin, “Perfectly secure steganography: Capacity, error exponents, and code constructions,” IEEE Trans. Inf. Theory, vol. 54, no. 6, pp. 2706–2722, Jun. 2008.

[26] A. D. Ker, “Steganalysis of LSB matching in grayscale images,” IEEE Signal Process. Lett., vol. 12, no. 6, pp. 441–444, Jun. 2005.

[27] M. Kharrazi, H. T. Sencar, and N. Memon, “Image steganography: Concepts and practice,” Lecture Notes Ser. Inst. Math. Sci., vol. 3, pp. 1–23, 2004.

[28] T. Morkel, J. H. P. Eloff, and M. S. Olivier, “An overview of image steganography,” Proc. ISSA, pp. 1–11, 2005.

[29] N. F. Johnson and S. Jajodia, “Exploring steganography: Seeing the unseen,” Computer, vol. 31, no. 2, pp. 26–34, Feb. 1998.