[Cyber Security and Applications 2 (2024) 100028](https://doi.org/10.1016/j.csa.2023.100028)

Contents lists available at [ScienceDirect](http://www.ScienceDirect.com/)

Cyber Security and Applications

journal homepage: <http://www.keaipublishing.com/en/journals/cyber-security-and-applications/>

Increasing embedding capacity of stego images by exploiting edge pixels in prediction error space

Habiba Sultana[a](#_bookmark0),[∗](#_bookmark2), A.H.M. Kamal[a](#_bookmark0), Tasnim Sakib Apon[b](#_bookmark1), Md. Golam Rabiul Alam [b](#_bookmark1)

a *Jatiya Kabi Kazi Nazrul Islam University, Mymensingh, 2220, Bangladesh*

b *BRAC University, Dhaka, 1212, Bangladesh*

a r t i c l e i n f o a b s t r a c t

*Keywords:*

Edge detection Predictor Embedding capacity PSNR

Steganography Steganalysis

In the field of data concealing, edge detection techniques are frequently employed, particularly for improving image quality and data security. These methods, however, have a lower embedding capacity. In order to take advantage of more edge pixels, many strategies are used nowadays. These schemes either combine the output from multiple edge detectors or enlarge the edges of an edge image by dilating. Even so, if the amount of data is vast, the techniques might not be able to conceal all of it. Therefore, a novel strategy for edge exploitation is still needed to regulate the effectiveness of edge detection-based data-hiding strategies. By using edge detectors in the prediction error space, we utilized more edge pixels in this study (PES). Applying a predictor on the cover image and then calculating the prediction errors, we prepared the PES. The edges in PES were then marked using the edge detector. The edge-error corresponding pixels received more information than the relevant pixels that

more edges, which does help to achieve a higher embedding capacity. We implanted *𝑥* number of secret bits in did not create an edge-error. Additionally, we combined the results from different edge detectors to produce edge pixels and *𝑦* number of bits in non-edge pixels where *𝑥 > 𝑦*. The simulation results show that the proposed

scheme outperforms its rivals on all performance-measuring criteria, including payload, stego image quality, and resistance to attack.

# Introduction

Security is an essential part of data communications. There are many techniques in the literature that are currently being used for securing the data. The premier one is cryptography. A cryptography method encrypts the secret messages into an unintelligible format, known as ciphertext, to prevent intruders from realizing the meaning of it [[1]](#_bookmark57). Such meaningless random content in a message, in fact, makes the in- truder curious and deeply attentive to look for some hidden meaning in it. Watermarking is another way of securing the message [[2]](#_bookmark58). How- ever, that technique is mainly used for serving data integrity. Addition- ally, some watermarking methods remain their masks visible on the cover media. Due to those limitations, steganography has taken a firm stand in the field of secured communication. Steganography is a pro- cess of hiding data in a cover media, e.g., image, audio, video, text, bio-signals, DNA sequence, etc. [[3]](#_bookmark59). Among these media, images are gaining popularity as a cover media for their higher degree of informa- tion redundancy and flexible size for communication over the Internet [[4]](#_bookmark60). In steganography, an embedding technique implants secrets in a cover media. The modified media is called stego media. That stego me- dia is then sent to a destination. The destination end applies an exact de-

∗ Corresponding author.

*E-mail address:* [srity.cse@jkkniu.edu.bd](mailto:srity.cse@jkkniu.edu.bd) (H. Sultana).

embedding algorithm to extract the secrets from it. Some de-embedding algorithms, additionally, generate the cover media from the stego one [[5–17]](#_bookmark61). Therefore, all of these methods can be divided into reversible or irreversible depending on their ability to build cover media from stego one.

Intruders can even check if there is something secret in it. In that case, they deal with the possibility of changing cover contents. Such analyses are known as steganalysis techniques [[18]](#_bookmark64). With these consid- erations in mind, the performance of steganography is evaluated with the ability to increase the data hiding rate, decrease the distortion rate of the carrier and resist attacks.

There are diverse techniques for hiding information in image con- tent. A major part of such algorithms applies edge detection methods [[1,2,19–32]](#_bookmark57), as a part of their embedding technique. These schemes find edge and non-edge pixels first and then embed more bits in edge pixels than non-edge ones; because the human visual system is less sensitive to changes in sharp areas of images compared to smooth areas. Such methodology is very useful in yielding both better stego quality and higher data security. However, the embedding capacity depends more on the number of identified edge pixels. The embedding capacity goes high when the number of edge pixels is large.

<https://doi.org/10.1016/j.csa.2023.100028>

Received 15 May 2023; Received in revised form 9 July 2023; Accepted 3 August 2023

Available online 9 August 2023

2772-9184/© 2023 The Authors. Publishing Services by Elsevier B.V. on behalf of KeAi Communications Co., Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

In 2016, Lee et al. [[19]](#_bookmark65) proposed a reversible data hiding method based on reduplicated exploiting modification, image interpolation, and canny edge detection. To enhance the image quality, Kumar et al.

[[20]](#_bookmark66) proposed a steganography system with fuzzy edge identification and least significant bit (LSB) substitution methods in 2018. S. Sun in 2015 showed another post-processing technique to improve the visual quality of stego images [[21]](#_bookmark67). He applied Huffman encoding to Canny- detected edge pixels to measure the number of bits that could be im- planted in each pixel. The method next used a correction method for increasing the visual quality of their stego image. Vanmathi and Prabu in 2017 employed fuzzy logic edge detection method for detecting edge pixels. The authors applied a chaotic method for encrypting messages and the LSB substitution technique for implanting those in edge pix- els [[22]](#_bookmark68). G. Swain in 2015 adopted a pixel value difference policy to enhance data security [[23]](#_bookmark69). Uses of Canny, Sobel, and fuzzy as edge de- tection were compared in [[24]](#_bookmark70) in 2017 by Bai et al.. Setiadi in 2022

[[25]](#_bookmark71) increased the number of edge pixels by applying morphological operator dilation on edge image owing to improve the edge pixels and thus, the embedding capacity. The author also applied hybrid edge de- tectors for the same purpose. In addition to the edge detection tech- nique, Kusuuma et al. [[26]](#_bookmark72) in 2018 applied an encryption method for the secret message to strengthen the security of hidden data. Khan et al.

[[27]](#_bookmark73) in 2015 replaced four LSBs of detected edge pixels in their data hiding algorithm. That scheme, indeed, destroyed the image quality roughly. Again, Chen et al. [[28]](#_bookmark74) in 2010 combined the results of Canny and Fuzzy edge detectors to exploit more edge pixels. Tseng and Leng in 2014 [[29]](#_bookmark75) improved the distortion rate of [[28]](#_bookmark74) by working on im- age blocks rather than the whole image at a time. Link in [[25]](#_bookmark71), Gaurav and Ghanekar in 2018 [[30]](#_bookmark76) applied a dilation operator to increase the number of edge pixels. They implanted secret bits by applying bit-wise exclusive-OR (XOR) operation between the binaries of each edge pixel and a message chunk. Vishnu et al. in 2020 [[31]](#_bookmark77) also used the Canny edge detection method to identify the edge pixels. Next, they computed pixel value differences between each of the two consecutive edge pix- els reading in a sequential manner. That pixel value differences were used to implant secret bits in edge pixels. Sultana and Kamal in 2021

[[32]](#_bookmark78) employed multiple edge detectors and then measured the desired edge pixels for data hiding by doing a logical AND operation among the pixels of each of the same positions in edge images. That policy is ef- fective for getting better stego quality while the size of the demanded implantable message is small. Setiadi et al. in 2018 [[32]](#_bookmark78), combined the results of canny and sobel edge detectors to increase the number of edge pixels.

Studies in the state of the art reveal that edge detection-based em- bedding schemes suffer from less embedding capacity. Therefore, as the demand for data hiding increases, so does the challenge with the capa- bility of such schemes, because the number of edge pixels obtained in such a conventional way may be unable to conceive the demanded in- formation. Such good methods will then be ineffective. Our target is to

The key contributions of this research are as follows:

* An image steganography approach has been proposed for eﬃcient data hiding on the edges of an image. We have applied edge detectors in prediction error space to increase the embedding capacity of the stego image which signifies the novelty of this research.
* We have adapted the technique of combining the results of multiple edge detectors owing to increase the number of edge pixels, further. We have also made our scheme enable to concealment of a different number of bits in edge and non-edge pixels. Therefore, the proposed scheme increases both the embedding capacity and the visual quality of stego images. At the same time, it shows strong resistance against statistical attacks.

Throughout this manuscript, [Section 2](#_bookmark3) discussed previous existing studies related to this manuscript. [Section 3](#_bookmark7) presents the proposed method neatly. The simulated results of our scheme are demonstrated in [Section 4](#_bookmark19). [Section 4.6](#_bookmark56) is devoted to testing the robustness of the pro- posed scheme against attacks. [Section 5](#_bookmark62) narrates the implications and limitations of our work. Finally, [Section 6](#_bookmark63) draws a conclusion of the article.

# Related works

Human visioning systems are sensitive to local changes. In the image, local changes are marked as edges. Edges in the image appear along ei- ther horizontal, vertical, or diagonal gradient directions. A filter, known as a kernel, is used to detect the edges in an image. Very commonly used edge detectors are Canny, Sobel, Log, Prewitt, Laplacian, and Fuzzy edge detectors. Now briefly describe those operators in the following:

* Roberts edge detection

Lawrence Roberts develops the Roberts edge detection (1965). It measures the 2-D spatial gradient of an image in a straightforward and quick manner.

|  |  |
| --- | --- |
| −1 | 0 |
| 0 | +1 |

*𝑅𝐺𝑥*

|  |  |
| --- | --- |
| 0 | −1 |
| +1 | 0 |

*𝑅𝐺𝑦*

This technique highlights areas of high spatial frequency, which fre- quently coincide with edges. The most common application of this approach is when both the input and the output are grayscale im- ages. The estimated full amplitude of the spatial gradient of the input image at each place in the output is represented by the pixel values at each location.

Robert’s cross-operator consists of 2 ×2 convolution kernels. *𝐺𝑥* is a

simple kernel and *𝐺𝑦* is rotated by 900. The gradient magnitude is:

√

graphic method. For this, we have enriched the edge detection space so |*𝐺*| = *𝑅𝐺*2 + *𝑅𝐺*2 enhance the embedding capacity of the edge detection-based stegano-

*𝑥*

*𝑦*

(1)

that the applied edge detector can detect more edge pixels. To get that, we applied a predictor on the image pixels. We have measured the pre- diction errors as well. In the prediction error space, we have applied the edge detector. That strategy noticeably increases the number of edge- forming errors in the error space. Corresponding positions of edge and non-edge errors in error space are noted. We have used that positional map to link to the image domain. That mapping is done to classify the image contents as edge and non-edge pixels. We then implant secrets in these pixels by embedding rules. That strategy boosts up the embedding capacity notably. Nevertheless, still, the scheme faces problems in hid- ing large volumes of data. We further try to increase the number of edge pixels by combining the edge images of multiple edge detectors. Exper- imental results deduce that the proposed method performs better than the other competing schemes, and shows a noticeable improvement in its capability of hiding more data.

An approximate magnitude is computed:

*𝐺* = *𝑅𝐺𝑥* + *𝑅𝐺𝑦* (2)

| | | | | |

* Sobel edge Detection

|  |  |  |
| --- | --- | --- |
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

*𝑆𝐺𝑥*

|  |  |  |
| --- | --- | --- |
| −1 | 0 | +1 |
| −2 | 0 | +2 |
| −1 | 0 | +1 |

*𝑆𝐺𝑦*

In 1970, Sobel developed the Sobel edge detection method. In order to highlight areas of high spatial frequency that correlate to edges,

the Sobel approach performs a 2 − *𝐷* spatial gradient quantity on

The gradient magnitude is:

|*𝐺*| = √*𝐿𝐺*2 + *𝐿𝐺*2

*𝑥*

*𝑦*

(7)

gradient magnitude at each location in *𝑛* input grayscale images. an image. Typically, it is used to determine the predicted absolute

least is made up of two 3 ×3 complication kernels. The other kernel According to the bottom table’s disclosure, the operator at the very is just the first one rotated 900. The Roberts Cross operator and this are extremely similar. Sobel Edge detection operator consists of 3 ×3 convolution kernels. *𝐺𝑥* is a simple kernel and *𝐺𝑦* is rotated by 900.

Canny method to locate image edges, it is crucial to remove noise

An approximate magnitude is computed:

*𝐺* = *𝐿𝐺𝑥* + *𝐿𝐺𝑦* (8)

| | | | | |

Typically, the Laplacian is employed to determine whether a pixel is on the light or dark side of an edge.

* + Canny Edge Detection The Canny edge detection method is one of

the common edge detection methods in the industry. It was first de-

The gradient magnitude is:

|*𝐺*| = √*𝑆𝐺*2 + *𝑆𝐺*2

*𝑥*

*𝑦*

(3)

signed in 1983 by John Canny for his master’s thesis at MIT, and it still outperforms several more recent algorithms. Prior to using the

An approximate magnitude is computed:

*𝐺* = *𝑆𝐺𝑥* + *𝑆𝐺𝑦* (4)

| | | | | |

* Prewitt edge detection

Prewitt first suggested the Prewitt edge detection in 1970. Prewitt is a reliable method for determining an edge’s size and direction.

This gradient-based edge detector is estimated in the 3 ×3 neighbor-

hood for eight directions. Calculations have been made for all eight

convolution masks. The next step is to choose one complexity mask, specifically with the largest module in mind.

The gradient magnitude is:

from the image. The Canny approach is superior since it applies the tendency to identify edges and a serious threshold value after pre- serving the characteristics of the edges in the image. These are the algorithmic steps:

* A Gaussian filter is used to remove noise from the input image.
* Calculating the gradient of image pixels in order to determine magnitude along the x and y dimensions.
* Suppress the non-max edge contributor pixel points while taking into account a cluster of neighbors for any curve pointing in the direction of the specified edge.
* As a last step, apply the hysteresis thresholding technique to pre-

|*𝐺*| = √*𝑃 𝐺*2 + *𝑃 𝐺*2

*𝑥*

*𝑦*

(5)

serve pixels with gradient magnitudes greater than or equal to 1

and exclude pixels with gradient magnitudes less than or equal

An approximate magnitude is computed:

*𝐺* = *𝑃 𝐺𝑥* + *𝑃 𝐺𝑦* (6)

| | | | | |

Prewitt detection tends to provide somewhat noisier findings but is computationally slightly easier to execute than Sobel detection.

|  |  |  |
| --- | --- | --- |
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| +1 | +1 | +1 |

*𝑃𝐺𝑥*

|  |  |  |
| --- | --- | --- |
| −1 | 0 | +1 |
| −1 | 0 | +1 |
| −1 | 0 | +1 |

*𝑃𝐺𝑦*

* LoG edge detection

A 2-D isotropic measurement of an image is the Laplacian of Gaus- sian. Laplacian, which is also utilized for edge detection, is the region of an image that is emphasized and experiences fast intensity vari- ations. To lessen the sensitivity of noise, the Laplacian is applied to an image that has been smoothed using a Gaussian smoothing filter. This operator creates a single grayscale image from a single grayscale image that is input. The input image is represented in Laplacian as a collection of discrete pixels. In order to approach the second deriva- tives in the definition, a discrete convolution kernel is discovered. The following are the top two kernels:

|  |  |  |
| --- | --- | --- |
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

*𝐿𝐺𝑥*

|  |  |  |
| --- | --- | --- |
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

*𝐿𝐺𝑦*

to 0.

Generally, edge detectors are used in pattern recognition, image mor- phology, and feature extractions. In the field of image steganography, edge detectors are used to improve the security of hide data as well as to collect the data hiding features. These schemes first detect the edge and non-edge pixels in a cover image. Then these schemes either im- plant in edge pixels only or embed a different number of bits in edge and non-edge pixels. Bearing that in mind, we studied a good number of articles on that state of the art. Among those, we found the works of Junlan Baiet al. [[24]](#_bookmark70) and Setiadi D. R. I. M. [[25]](#_bookmark71) which were very close to our research objectives. As a result, we studied these two articles with deep attention and built the foundation of our proposed work on them. In 2017, Junlan Bai et al. [[24]](#_bookmark70) proposed a steganography approach based on the combination of edge detection and LSB substitution meth- ods. In their scheme, the authors first prove that if a few of the LSBs of image pixels are cleared and then an edge detector is applied on that

cleared image, let *𝐼𝑐 𝑙𝑟*, the detection information is almost the same as

age, say *𝐼𝑜*. Therefore, they proposed to clear *𝑛* bits of information in an computing edge information by applying the detector in the original im-

in the pixels of the cover image by *𝑘* − *𝐿𝑆𝐵* substitutions, where *𝑘* ≤ *𝑛*. image before applying an edge detector. They are allowed to implant In practice, they used *𝑛* = 5. Thereafter, they formed stego image *𝐼𝑠* by implanting *𝑥*-bits of information in detected edge pixels and *𝑦*-bits of in- formation in non-edge pixels, where *𝑥 > 𝑦*. At the extraction phase, their scheme again cleared *𝑛* bits of LSBs from stego pixels of *𝐼𝑠*. That cleared image is, indeed, *𝐼𝑐 𝑙𝑟*. Hence, the extraction phase feels no ambiguity to

then, extracts *𝑥* bits of LSBs from edge pixels and *𝑦* bits of LSBs from detect the same set of edge pixels as the data-hiding phase. The scheme,

non-edge pixels.

ilar to Junlan Bai et al.’s scheme. He also implanted *𝑥* bits of informa- In 2019, Setiadi proposed an image steganography that is very sim- tion in edge pixels and *𝑦* bits of secrets in non-edge pixels. However, he

added some pre-processing tasks to increase the number of edge pixels

detectors, say *𝑚* number of detectors, on 5-LSBs cleared images. These owing to improving the embedding capacity. He applied multiple edge

*𝑚* detectors separately generate *𝑚* edge images. Let these edge images are *𝐼𝑒*1, *𝐼𝑒*2, ..., *𝐼𝑒𝑚*. The scheme increases the number of edge pixels by

combining these edge results by doing logical OR operations among the

**Table 1**

Summary of related works.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criteria | Bai [[24]](#_bookmark70) | Setiadi [[25]](#_bookmark71) | Sultana [[32]](#_bookmark78) | Rasol [[34]](#_bookmark80) | Proposed System |
| Use an edge detector? | Yes | Yes | Yes | Yes | Yes |
| Encrypt message? | No | Yes | No | No | No |
| Cleared LSBs | *𝑛*-bits | 5-bits | *𝑛*-bits | No | *𝑛*-bits |
| Use multi predictors | No | Yes | Yes | Yes | Yes |
| Hybridize edge image? | No | Yes | Yes | Yes | Yes |
| Dilate edge image? | No | Yes | No | No | No |
| Embed as (*𝑥, 𝑦*) bits? | Yes | Yes | Yes | Yes | Yes |
| Message type? | Binary | Text | Binary | Text | Binary |
| Use Prediction Error space? | No | No | No | No | Yes |

edge images, i.e., by *𝐼𝑒*1 ∨ *𝐼𝑒*2 ∨…∨ *𝐼𝑒𝑚*. Let the resultant image is *𝑟𝐼𝑒*. He dilated the edge image *𝑟𝐼𝑒* to increase edge pixels more. He then

separated the edge and non-edge pixels from the dilated image.

In 2021, Sultana et al. proposed image steganography based on hy- brid edge detectors. In their scheme, the authors applied multiple edge

detectors, say *𝑝* number of detectors, on *𝑚*− LSBs cleared image, and

those detectors separately generated edge images. These edge images

are combined using logical AND operations. The authors then implant

where function Ψ returns the remainder value when one divides *𝐼* (*𝑖, 𝑗*)

by 2*𝑛*.

* + 1. *Generating prediction error space*

Let the height and width of image *𝐼* be *ℎ* and *𝑤*, respectively. Then,

the predicted values of four corner pixels are computed by [Eq. (10)](#_bookmark5).

⎧⎪ *𝑝*(1*,* 1) = *𝐼* (1*,*2)+*𝐼* (2*,*1)

2

⎪

*𝑥* bits into edge pixels and *𝑦* bits into non-edge pixels, where *𝑥 > 𝑦*.

In 2018, Setiadi proposed image steganography based on canny-

Sobel edge detector and LSB substitution methods. In this scheme, the

⎪*𝑝*(1*, 𝑤*) =

⎨

*𝐼* (1*, 𝑤* − 1) + *𝐼* (2*, 𝑤*) 2

*𝐼* (*ℎ* − 1*,* 1) + *𝐼* (*ℎ,* 2)

(10)

authors applied multiple edge detectors, say *𝑚* number of detectors, on the cover image. These *𝑚* detectors separately generate *𝑚* edge images. Let these edge images be *𝐼*1, *𝐼*2, ..., *𝐼𝑚*. The scheme hybridized those edge images using logical OR operations, i.e., by *𝐼*1 ∨ *𝐼*2 ∨…∨ *𝐼𝑚*. Let the resultant image is *𝑟𝐼𝑒*. As a result, this scheme increases the number

of edge pixels. At the same time, the author is added one special char-

into binary form. The authors then implant *𝑥* bits into edge pixels and acter at the end of the secret message then this message is transformed

*𝑝*(*ℎ,* 1) =

2

*𝑝*(*ℎ, 𝑤*) = *𝐼* (*ℎ* − 1*, 𝑤*) + *𝐼* (*ℎ, 𝑤* − 1)

⎩⎪

⎪

2

Similar way, the other pixels in the first row, last row, first column, and last column are calculated by [Eq. (11)](#_bookmark6).

⎧⎪ *𝑝*(1*, 𝑗*) = *𝐼* (1*, 𝑗* − 1) + *𝐼* (2*, 𝑗*) + *𝐼* (1*, 𝑗* + 1)

3

*𝑦* bits into non-edge pixels (where *𝑥 > 𝑦*) based on message length.

*𝐼* (*ℎ, 𝑗* − 1) + *𝐼* (*ℎ* − 1*, 𝑗*) + *𝐼* (*ℎ, 𝑗* + 1)

*𝑝 ℎ, 𝑗*

⎪ ( ) =

[Table 1](#_bookmark4) provides a summary of comparisons among the existing stud- ies. It is also demonstrated how our proposed model differs from previ- ous works.

⎪⎨ *𝑝*(*𝑖,* 1) =

⎪

⎪*𝑝*(*𝑖, 𝑤*) = *𝐼* (*𝑖* − 1*, 𝑤*) + *𝐼* (*𝑖, 𝑤* − 1) + *𝐼* (*𝑖* + 1*, 𝑤*)

⎩ 3

3

*𝐼* (*𝑖* − 1*,* 1) + *𝐼* (*𝑖,* 2) + *𝐼* (*𝑖* + 1*,* 1)

3

(11)

# Proposed method

The proposed work consists of three phases: pre-processing, data em- bedding, and data extracting. Our Initial phase of pre-processing is ex- plained in [Section 3.1](#_bookmark10). Data embedding is present in [Section 3.2](#_bookmark17) and

In the same way, we compute all other middle pixels by [Eq. (12)](#_bookmark8).

*𝑝*(*𝑖, 𝑗*) = *𝐼* (*𝑖* − 1*, 𝑗*) + *𝐼* (*𝑖, 𝑗* − 1) + *𝐼* (*𝑖* + 1*, 𝑗*) + *𝐼* (*𝑖, 𝑗* + 1)

4

(12)

finally [Section 3.3](#_bookmark24) goes over our extraction phase. [Figs. 1](#_bookmark14) and [2](#_bookmark15) depict the overall proposed method.

* 1. *Pre-processing phase*

As our target is to implant more bits in edge pixels than the non- edges, both the data hider and the data extractor should detect and iden- tify the same set of edge pixels. To do that equally, a homogeneous field is required where both the data hider and data extractor can detect the same set of edge pixels. Previous works commonly applied their edge detectors in that homogeneous field. But, in that stage, our target is to apply a predictor on that homogeneous field and, thereafter, to compute the prediction errors. We would then want to compute edge pixels with the help of that prediction error space. These pre-processing tasks are shown in blocks of the first four levels of [Fig. 1](#_bookmark14) Encoder.

* + 1. *Generating homogeneous field*

Before starting that discussion, let the taken cover image is *𝐶* and an instance of it is *𝐼* . We first clear *𝑛*-bits of LSBs from every pixel of

*𝐼* . To explain the LSB clearing method, consider a pixel value at (*𝑖, 𝑗*)

location of *𝐼* , i.e., *𝐼* (*𝑖, 𝑗*), is 175. The binary conversion of that pixel

value is 01001011. Then, clearing 3 bits of LSBs from 01001011 will

of *𝑛*-bits of LSBs from each pixel by [Eq. (9)](#_bookmark13). yield 01001000. In decimal, it will be 72. We generalize that clearing

*𝐼* (*𝑖, 𝑗*) = *𝐼* (*𝑖, 𝑗*) − Ψ(*𝐼* (*𝑖, 𝑗*)*,* 2*𝑛*); (9)

Thus, we compute all predicted values. Finally, the prediction errors and the absolute values of the prediction errors are calculated by [Eqs. (13)](#_bookmark9) and [(14)](#_bookmark11), respectively.

*𝑝𝐸*(*𝑖, 𝑗*) = *𝐼* (*𝑖, 𝑗*) − *𝑝*(*𝑖, 𝑗*) (13)

*𝑝𝐸𝑎*(*𝑖, 𝑗*) = *𝑝𝐸*(*𝑖, 𝑗*) (14)

| |

where 1 ≤ *𝑖* ≤ *ℎ*, 1 ≤ *𝑗* ≤ *𝑤* and *𝑎* stands for absolute value of *𝑎*.

| |

* + 1. *Edge image generation*

We have applied *𝑚*-number of ‘edge detection operators’, e.g., canny, sobel, fuzzy, Robert, Prewitt, log, etc., on *𝑝𝐸𝑎* to measure edge pixels,

separately. We have generated the edge image by [Eq. (15)](#_bookmark12).

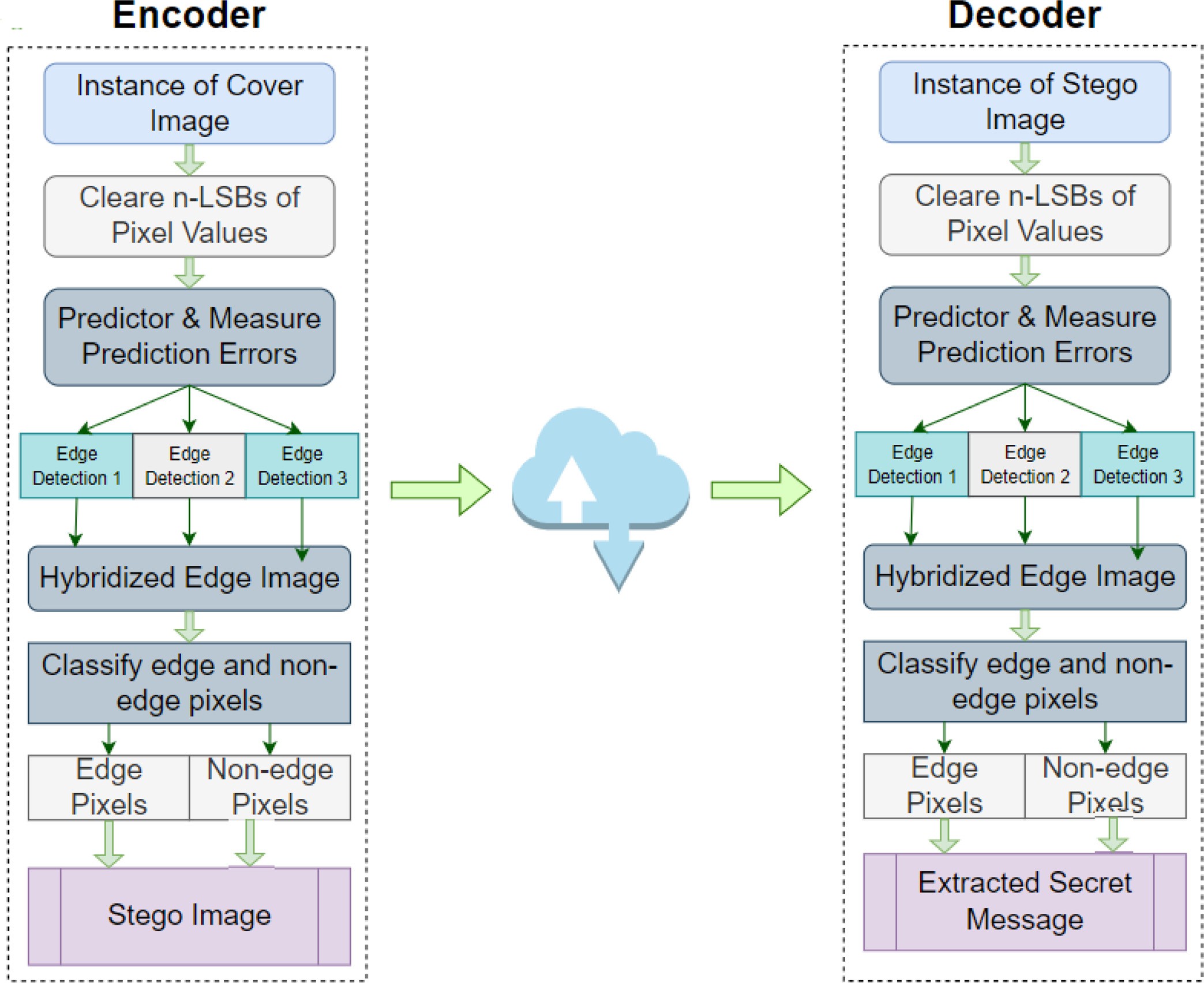
*𝑒𝐼* (*𝑖*) = *𝑓* (*𝑝𝐸𝑎,* Ω) (15)

where Ω is one of the *𝑚* edge detection operators, i.e., Ω ∈ {*𝐶𝑎𝑛𝑛𝑦*,

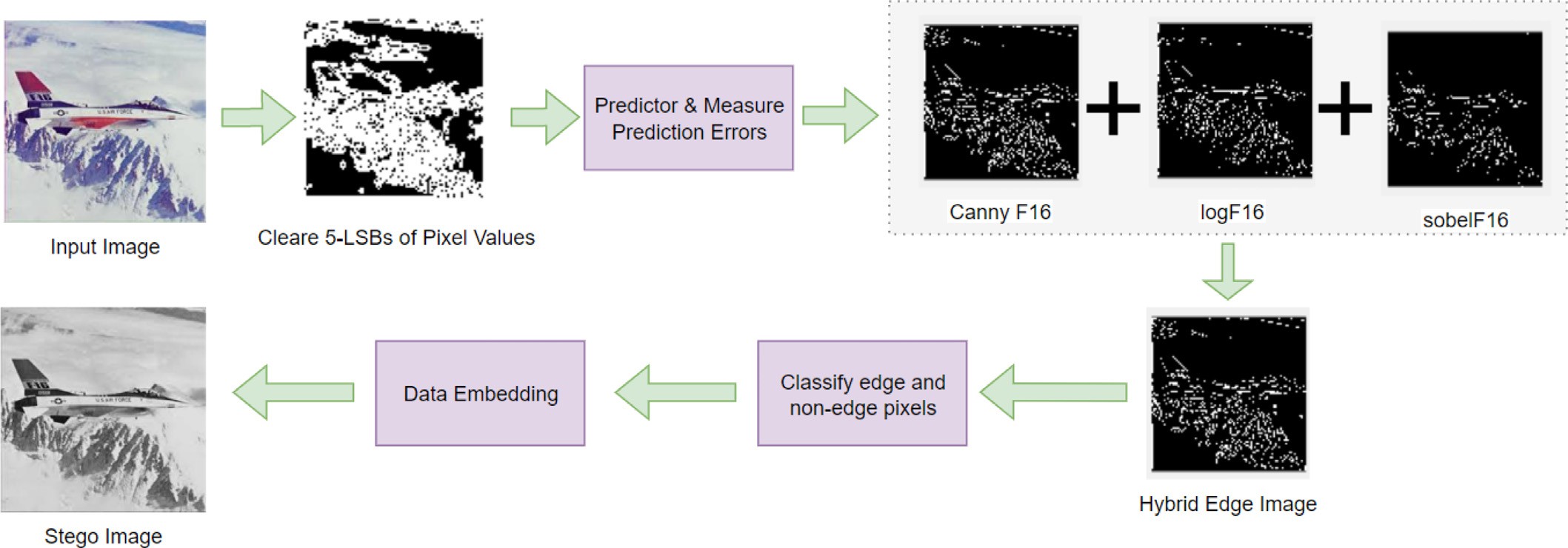
*𝑆𝑜𝑏𝑒𝑙, 𝐿𝑜𝑔*, *𝐹 𝑢𝑧𝑧𝑦, 𝑅𝑜𝑏𝑒𝑟𝑡*, *𝑃 𝑟𝑒𝑤𝑖𝑡𝑡, 𝑒𝑡𝑐.*}, 1 ≤ *𝑖* ≤ *𝑚* and *𝑓* returns the edge image *𝑒𝐼* from *𝑝𝐸𝑎* for a specific edge detector Ω. Each edge image

is a binary image. For each pixel, the edge image holds a 0 or 1. A 1 in

image *𝑝𝐸𝑎* is in the detected edge. We consider that the edges in *𝑝𝐸𝑎* are edge image means the corresponding pixel (indeed, absolute error) of the edges of *𝐼* . The experiment reveals that the number of edge pixels in *𝑝𝐸𝑎* is more than that in *𝐼* for the same edge detector. Hence, the application of an edge detector in *𝑝𝐸𝑎* will provide us with more edge



**Fig. 1.** The Encoder-Decoder architectures of the Proposed Method.



**Fig. 2.** The top-level overview of the Proposed Method.

pixels in *𝐼* . Our objective is to enhance the embedding capacity of an

edge detection-based scheme. Therefore, such an attempt will lead us

to our goal. To improve further, we have combined the detected edge results. To do that, we have applied logical OR operations among the

edge images. The OR image, i.e., resultant image *𝑟𝐼* (*𝑖, 𝑗*), is computed

by [Eq. (16)](#_bookmark16).

*𝑟𝐼* (*𝑖, 𝑗*) = *𝑒𝐼* (1*, 𝑖, 𝑗*) ∨ *𝑒𝐼* (2*, 𝑖, 𝑗*) ∨…∨ *𝑒𝐼* (*𝑚, 𝑖, 𝑗*) (16)

where 1 ≤ *𝑖* ≤ *ℎ,* 1 ≤ *𝑗* ≤ *𝑤* and ∨ stands for logical OR operator. This

way, we have generated the resultant edge image from prediction error

space by hybridizing edge detectors.

values, respectively. The absolute values of prediction errors are *𝑝𝐸𝑎*. [Eqs. (13)](#_bookmark9) and [(14)](#_bookmark11) compute the prediction errors and their absolute Using [Eq. (15)](#_bookmark12) we have computed edge image *𝑒𝐼* (*𝑖*) from *𝑝𝐸𝑎* for dif- ferent edge detectors Ω. Combining all edge images by [Eq. (16)](#_bookmark16), we form the resultant edge image *𝑟𝐼* . We have separated the edge pixels

*𝑒𝑃* and their corresponding positions *𝑒𝑃 𝑃* in *𝐼* by [Eq. (17)](#_bookmark20). The same [Eq. (17)](#_bookmark20) helps us in finding non-edge pixels *𝑛𝑒𝑃* and their positions *𝑛𝑒𝑃 𝑃* in *𝐼* . Next from each of the edge-located pixels, i.e., from (*𝑠, 𝑡*), we have measured *𝑑𝑥*, where *𝑑𝑥* = *𝑆*(*𝑠, 𝑡*) − *𝐼* (*𝑠, 𝑡*). Similar way, from each of the non-edge located pixels, we have computed *𝑑𝑦* by *𝑑𝑦* = *𝑆*(*𝑢, 𝑣*) − *𝐼* (*𝑢, 𝑣*). Here, *𝑠, 𝑡, 𝑢, 𝑣, 𝑖*, and *𝑗* are defined in the immediate previous subsection.

Next, we have extracted the binaries of the secret by [Eq. (19)](#_bookmark18).

* 1. *Data implantation phase*

{*𝑏𝑥*

= Φ(*𝑑𝑥*)

(19)

Now, it is the turn to embed secrets in image *𝐼* . The whole data

hiding process is depicted in [Fig. 1](#_bookmark14) Encoder. The data-hiding steps are

as follows:

location in *𝐼* . That classification is done using the help of the resul- *•* A module is developed to classify edge and non-edge pixels and their tant edge image *𝑟𝐼* . The [Eq. (17)](#_bookmark20) is used to do that.

[*𝑒𝑃 , 𝑒𝑃 𝑃 , 𝑛𝑒𝑃 , 𝑛𝑒𝑃 𝑃* ] = *𝐹* (*𝐼 , 𝑟𝐼* ) (17)

where *𝐹* returns edge pixels *𝑒𝑃* , their positions *𝑒𝑃 𝑃* , non-edge pix- els *𝑛𝑒𝑃* and their positions *𝑛𝑒𝑃 𝑃* in *𝐼* . The function *𝐹* performs the

calculation of these four return values by the following module of

*𝐹* .

Function *𝐹* (*𝐼 , 𝑟𝐼* )

Compute the size of image *𝐼* . Let it is (*ℎ, 𝑤*) [*𝑒𝑃 , 𝑒𝑃 𝑃* ] = *𝐺𝑒* (*𝐼 , 𝑟𝐼 , ℎ, 𝑤*)

*𝑐𝑟𝐼* = (*𝑟𝐼* − 1) × (−1)

[*𝑛𝑒𝑃 , 𝑛𝑒𝑃 𝑃* ] = *𝐺𝑒* (*𝐼 , 𝑐𝑟𝐼 , ℎ, 𝑤*)

return [*𝑒𝑃 , 𝑒𝑃 𝑃 , 𝑛𝑒𝑃 , 𝑛𝑒𝑃 𝑃* ]

where *𝐺𝑒* is defined below.

Function *𝐺𝑒* (*𝐴, 𝐵, ℎ, 𝑤*)

*𝑘* = 0

for *𝑖* = 1 to *ℎ* for *𝑗* = 1 to *𝑤* if *𝐵*(*𝑖, 𝑗*) == 1

*𝑅*1(*𝑘*) = *𝐴*(*𝑖, 𝑗*)

*𝑅*2(*𝑘,* 1) = *𝑖, 𝑅*2(*𝑘,* 2) = *𝑗*

return *𝑅*1*, 𝑅*2

* Next, we implant x bits and y bits of secrets in each edge and non-

method. Let *𝑥*-bits of information is *𝑏𝑥* and *𝑦*-bits of information is edge pixel, respectively, of the cover image by the LSB substitution

*𝑏𝑦*. In that, the substitution task is performed by [Eq. (18)](#_bookmark22)

{

*𝑏𝑦* = Φ(*𝑑𝑦*)

where Φ(*𝑑𝑥*) means binary conversion of decimal value *𝑑𝑥* and Φ(*𝑑𝑦*)

means binary conversion of decimal value *𝑑𝑦*.

[Fig. 1](#_bookmark14) Decoder represents the overall extraction phase.

# Result analysis

This Section is divided into several sub-sections. In [Section 4.1](#_bookmark21) we go over the experimental setup with which our study is carried out. [Section 4.4](#_bookmark46) discusses the mathematical representation of feature values. Later in [Section 4.5](#_bookmark55) we analyze the complexity. [Section 4.2](#_bookmark23) reveals our experimental results. Finally, [Section 4.3](#_bookmark28) provides a clear discussion re- garding our findings.

We have investigated and compared the performance of the proposed scheme with the works of Bai et al. [[24]](#_bookmark70), Setiadi [[25]](#_bookmark71), Sultana et al.

[[32]](#_bookmark78) and Rasol et al. [[34]](#_bookmark80). We first selected images for the experiment, set up our experiment, and then analyzed the results.

* 1. *Experimental setup*

We worked on MATLAB’s edition R(2017a) on Windows 7. The con- figuration of the laptop was comprised of an Intel (R) Core (TM) i5- 8500T CPU@ 2.10 GHz 2.11 GHz processor and RAM of 8.00 GB. We first collected 10 commonly used images, as shown in [Fig. 3](#_bookmark26), as a stan- dard image dataset. We conducted all primary experiments on that stan- dard image dataset. To test our scheme in a large image dataset, we used the BOSS dataset [[33]](#_bookmark79). The BOSS dataset consisted of 499 images. We

resized the images to 512 × 512 and converted the color to grayscale to

fit our program.

We developed two different programs for each of the proposed meth- ods, Bai et al., Setiadi D. R. I. M., Sultana et al. and Rasol et al. where one is for data implantation and the other is for retrieving the implanted

secrets from stegos. According to the algorithm, we cleared *𝑛* bits of LSBs

*𝑆*(*𝑠, 𝑡*) = *𝐼* (*𝑠, 𝑡*) + Φ(*𝑏𝑥*)

*𝑆*(*𝑢, 𝑣*) = *𝐼* (*𝑢, 𝑣*) + Φ(*𝑏𝑦*)

(18)

from the cover images. We then verified the schemes by implanting dif- ferent numbers of bits in edge and non-edge pixels, however, these are

where Φ(*𝑏𝑥*) stands for decimal conversion of binary *𝑏𝑥*, *𝑠* =

*𝑒𝑃 𝑃* (*𝑖,* 1)*, 𝑡* = *𝑒𝑃 𝑃* (*𝑖,* 2), *𝑢* = *𝑛𝑒𝑃 𝑃* (*𝑗,* 1)*, 𝑣* = *𝑛𝑒𝑃 𝑃* (*𝑗,* 2) and 1 ≤ *𝑖* ≤

*𝑁𝑜*\_*𝑂𝑓* \_*𝐸𝑑𝑔𝑒*\_*𝑃 𝑖𝑥𝑒𝑙𝑠,* 1 ≤ *𝑗* ≤ *𝑁𝑜*\_*𝑂𝑓* \_*𝑛𝑜𝑛𝐸𝑑𝑔𝑒*\_*𝑃 𝑖𝑥𝑒𝑙𝑠*. Here, *𝑏𝑥* will be different for each of the *𝑠* and *𝑡*. The same is true for *𝑏𝑦*. This means that, each time a different *𝑏𝑥* and *𝑏𝑦* of secret will be implanted.

That stego image *𝑆* is then sent to a receiver end. The receiver end next extracts the implanted secrets from *𝑆*.

*3.3. Extraction phase*

The extraction phase undergoes necessary pre-processing tasks. The

receiver copies the stego image *𝑆* to *𝐼* . It then clears *𝑛* bits of LSBs extraction phase is depicted in [Fig. 1](#_bookmark14) Decoder. Like the sender, the from *𝐼* by [Eq. (9)](#_bookmark13). Consider, that *𝑛*-LSB cleared image is also *𝐼* . The scheme then applies the same predictor to *𝐼* to predict its pixel val-

ues. That prediction of pixel values is done by [Eqs. (10)](#_bookmark5) to [(12)](#_bookmark8). Again,

no more than *𝑛* bits in any way. The performance of the schemes is ver-

ified by several feature values, such as edge pixel generation capability,

embedding payload, peak signal-to-noise ratio (PSNR), structural simi- larity index matrix (SSIM), standard deviation, correlation coeﬃcient, entropy, cosine similarity, pixel difference histogram, and chi-square text, etc.

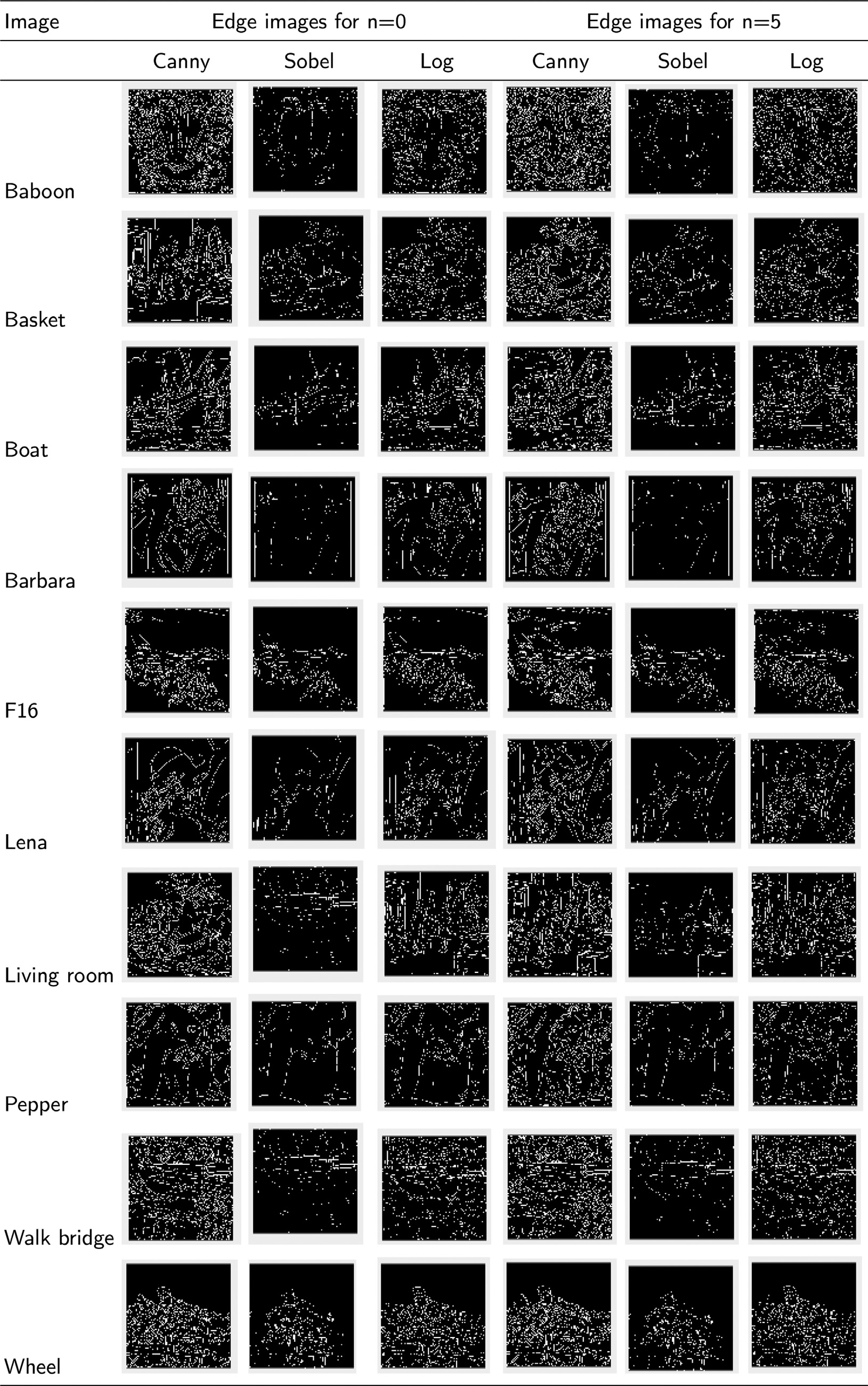
* 1. *Experimental results*

In order to distinguish between edge and non-edge pixels in the ex- periment, we initially employed Canny, Sobel, and Log edge detectors in one, three, and five LSBs cleared images. When given an input image, the Canny, Sobel, and Log-based edge detector functions of MATLAB return an edge image. Binary images are used for edges. Utilizing 10 input images as a starting point, [Table 2](#_bookmark25) depicts the edge images that

plant *𝑥* bits of information in edge pixels and *𝑦* bits of information in were generated. It is already mentioned that the embedding rules im-

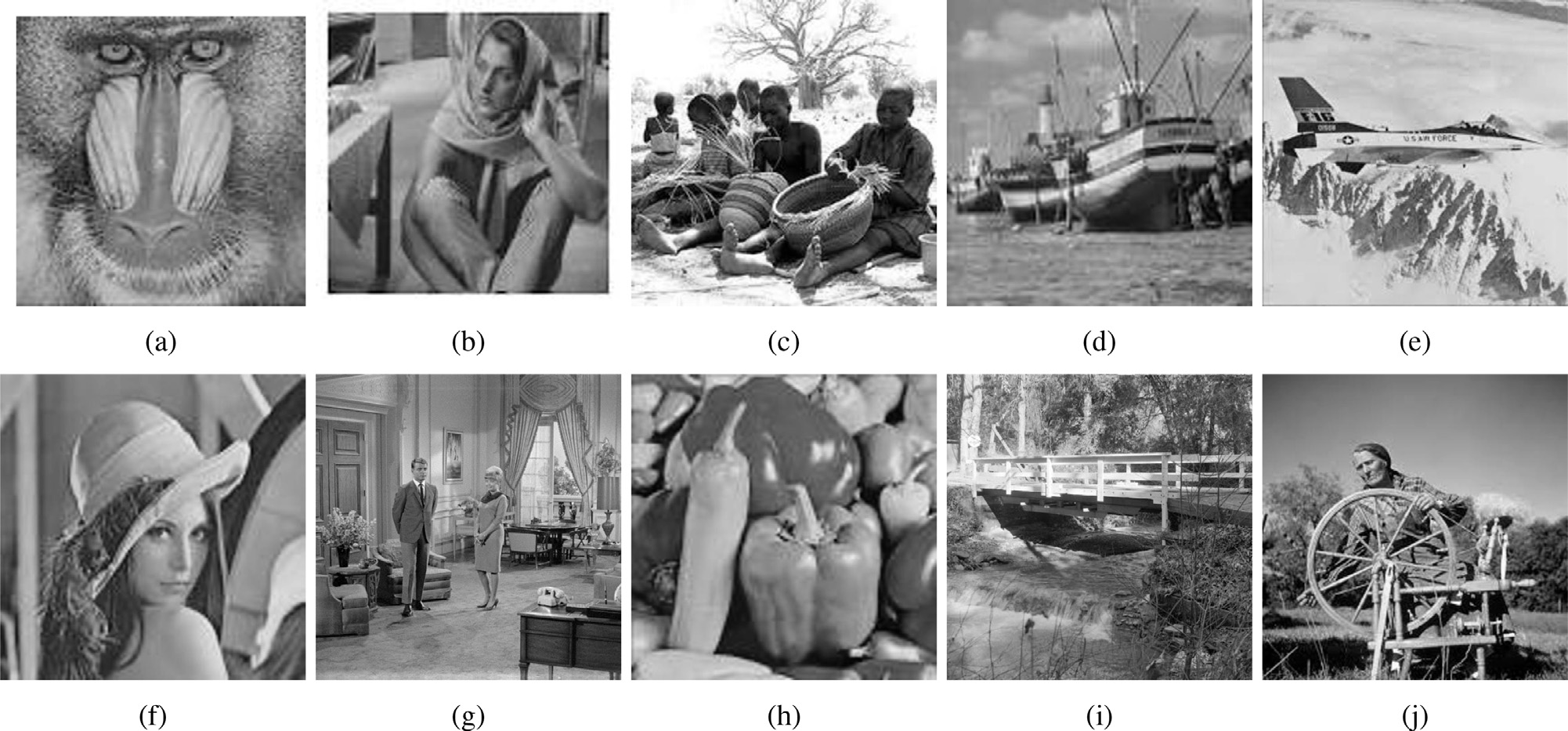
**Table 2**

original images (*𝑛* = 0) and 5-LSBs cleared images (*𝑛* = 5). Edge images generated from ten cover images. The images were formed for Canny, Sobel, and Log edge detectors both from





**Fig. 3.** Cover images for the experiment (**a**) Baboon, (**b**) Barbara, (**c**) Basket, (**d**) Boat, (**e**) F16, (**f**) Lena, (**g**) Livingroom, (**h**) Peppers, (**i**) Walkbridge, (**j**) Wheel.



**Fig. 4.** Stego images for the corresponding cover images: (**a**) Baboon, (**b**) Barbara, (**c**) Basket, (**d**) Boat, (**e**) F16, (**f**) Lena, (**g**) Livingroom, (**h**) Peppers, (**i**) Walkbridge,

(**j**) Wheel.

tuple (*𝑥, 𝑦*). As *𝑥 > 𝑦*, an attempt of increasing the number of edge pix- non-edge pixels. In all following discussions, we will represent that as a

els will certainly boost the yielded embedding capacity. We did it by detecting edges in prediction error space. The justification is shown in [Figs. 3](#_bookmark26) and [4](#_bookmark27). [Table 3](#_bookmark29) also summarises the number of edge pixels that were found in ten sample images by different methods. [Table 3](#_bookmark29) provided statistics collected from 5-LSBs cleared images.

We analyzed all the proposed and competing schemes to check their quantitative ability in hiding data. For that, all the schemes were exper-

imented with for different values of (*𝑥, 𝑦*). Here [Tables 5](#_bookmark51) and [6](#_bookmark51) repre-

sent the payload results for (*𝑥, 𝑦*) ∈ {(2*,* 1), (3*,* 1)*,* (3*,* 2), (4*,* 1)*,* (4*,* 2), (4*,* 3)}.

the different values of (*𝑥, 𝑦*) is published. We analyzed the performance There, a list of how much data can be hidden by a scheme for each of

shown in [Fig. 7](#_bookmark32) for (*𝑥, 𝑦*) = (4*,* 1). The same result is depicted in [Fig. 8](#_bookmark33) for in the embedding capacity as well. Embedding capacity is graphically

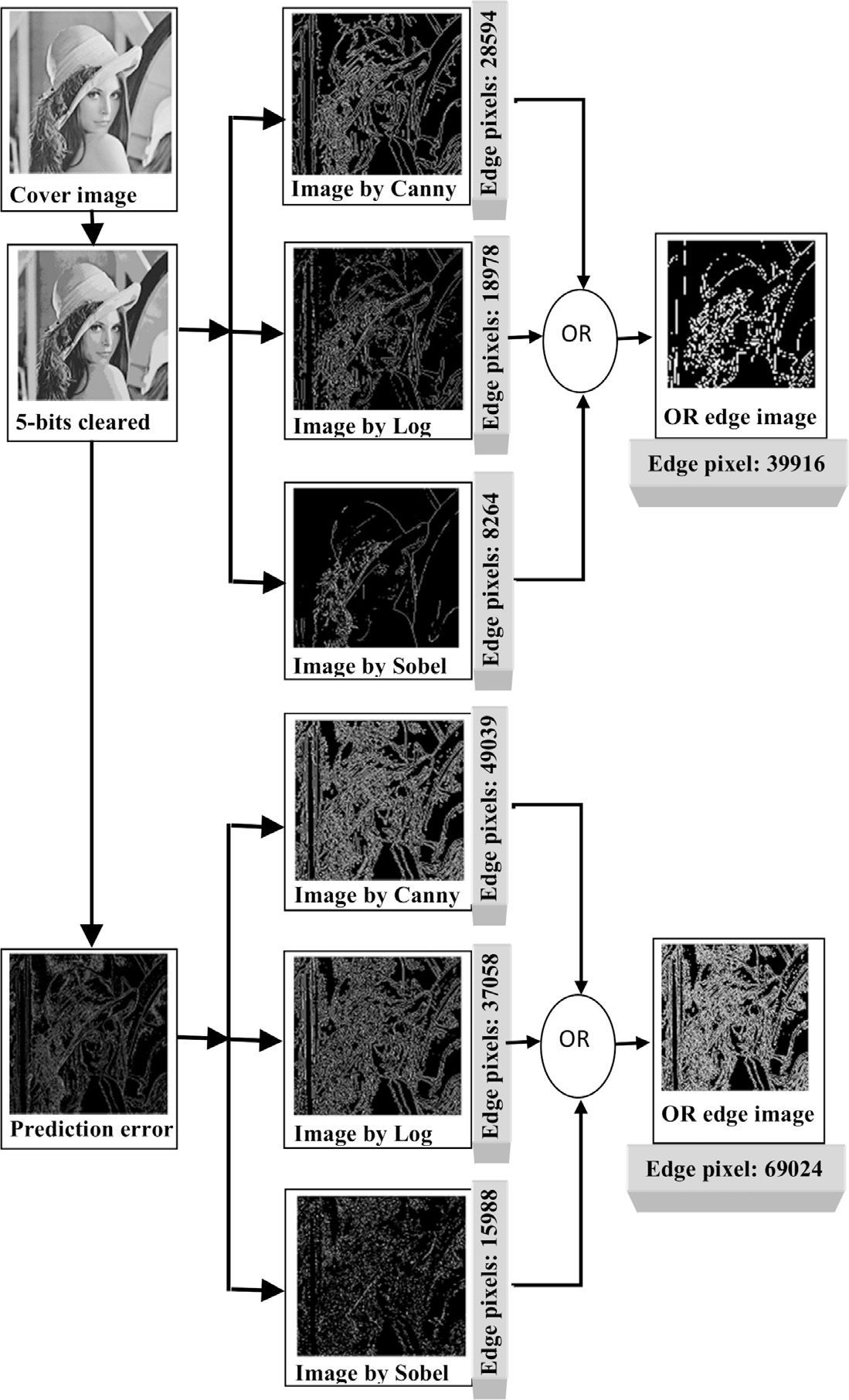
499 images of the BOSS dataset.

We also analyzed the visual quality and structural originality of stego images. Visual quality is measured by PSNR values. Payload per losses of

for every (*𝑥, 𝑦*) are listed in [Tables 7](#_bookmark52) and [8](#_bookmark53). PSNR is sketched in [Fig. 9](#_bookmark34). The structural similarity index value, SSIM,

* 1. *Discussions on results*

The primary objective of this research is to strengthen the data- hiding ability of our scheme in terms of implanted data amount. As we worked on edge detection-based data embedment arena and the pro- posed scheme implanted more bits in edge pixels than the non-edge ones, we gave emphasis to increasing the number of edge pixels. For this, we applied the edge detector algorithms in prediction error space. Experiments have shown that this method is quite effective. The results of [Fig. 5](#_bookmark30) is a snap-shoot of our experiment that is done on Lena’s im- age. The figure is annotated with obtained edge pixels. Compared to



**Table 3**

Statistics of Edge pixels for *𝑛* = 5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image | Methods |  |  |  |  |
| 512 × 512 | Proposed | Bai | Setiadi | Sultana | Rasol |
| F16 | 120,261 | 24,966 | 69,625 | 17,640 | 23,416 |
| babon | 176,798 | 38,383 | 100,727 | 4718 | 34,566 |
| basket | 159,679 | 31,560 | 86,141 | 21,975 | 31,223 |
| boat | 138,274 | 26,991 | 82,727 | 32,813 | 24,326 |
| brbra | 140,756 | 26,941 | 74,693 | 6272 | 18,521 |
| lena | 126,397 | 24,884 | 75,604 | 27,664 | 21,032 |
| livingroom | 186,518 | 35,543 | 102,571 | 27,118 | 36,722 |
| pepper | 136,479 | 25,860 | 80,497 | 31,724 | 19,259 |
| walkbridge | 199,069 | 45,563 | 123,388 | 24,133 | 44,923 |
| wheel | 142,915 | 27,745 | 75,688 | 21,585 | 29,419 |

state-of-art, that number is about double in our scheme. For ten images of the standard dataset, the results are shown in detail in [Fig. 5](#_bookmark30) and

[Table 3](#_bookmark29). [Table 3](#_bookmark29) states the results for *𝑛* = 5, where [Fig. 5](#_bookmark30) depicts the

same for *𝑛* = 1, *𝑛* = 3 and *𝑛* = 5. Corresponding embedding capacities are

depicted in [6](#_bookmark31). The results of [Figs. 5](#_bookmark30) and [6](#_bookmark31) and [Table 3](#_bookmark29) confirm that all

edge detectors recognize more contents as edge contents in prediction error space.

**Fig. 5.** A scene of how the edge detectors perform in prediction error space in detecting edge. The figure shows that each edge detector detects a large number of edge pixels in the prediction error space than in the image domain. Since our target is to increase the number of Edge pixels, this figure proves that the application of an edge detector in prediction error space has been successful.

edge pixels at a rate of (*𝑥, 𝑦*). The embedding capacity is depicted in After justifying that truth, we implanted secrets in edge and non-

[Fig. 7](#_bookmark32) for ten sample images. The figure states that the proposed scheme enhances the embedding capacity by about Bai 80% of et al. [[24]](#_bookmark70), 50% of Setiadi [[25]](#_bookmark71), 90% of Sultana et al. [[32]](#_bookmark78) and 70% of Rasol et al. [[34]](#_bookmark80). To justify the performance of the proposed scheme, we experi- mented on 499 images of the BOSS dataset as well. The experiments said the same fact. Those results are summarised in [Fig. 9](#_bookmark34) In all the im- ages, the proposed scheme provides dominating results over the other competing schemes. However, those depicted results were obtained for

(*𝑥, 𝑦*) = (4*,* 1) only. [Table 4](#_bookmark42) represents the results for all combinations

of (*𝑥, 𝑦*), e.g., (*𝑥, 𝑦*) ∈ {(2*,* 1)*,* (3*,* 1)*,* (3*,* 2)*,* (4*,* 1)*,* (4*,* 2)*,* (4*,* 3)}. Additionally,

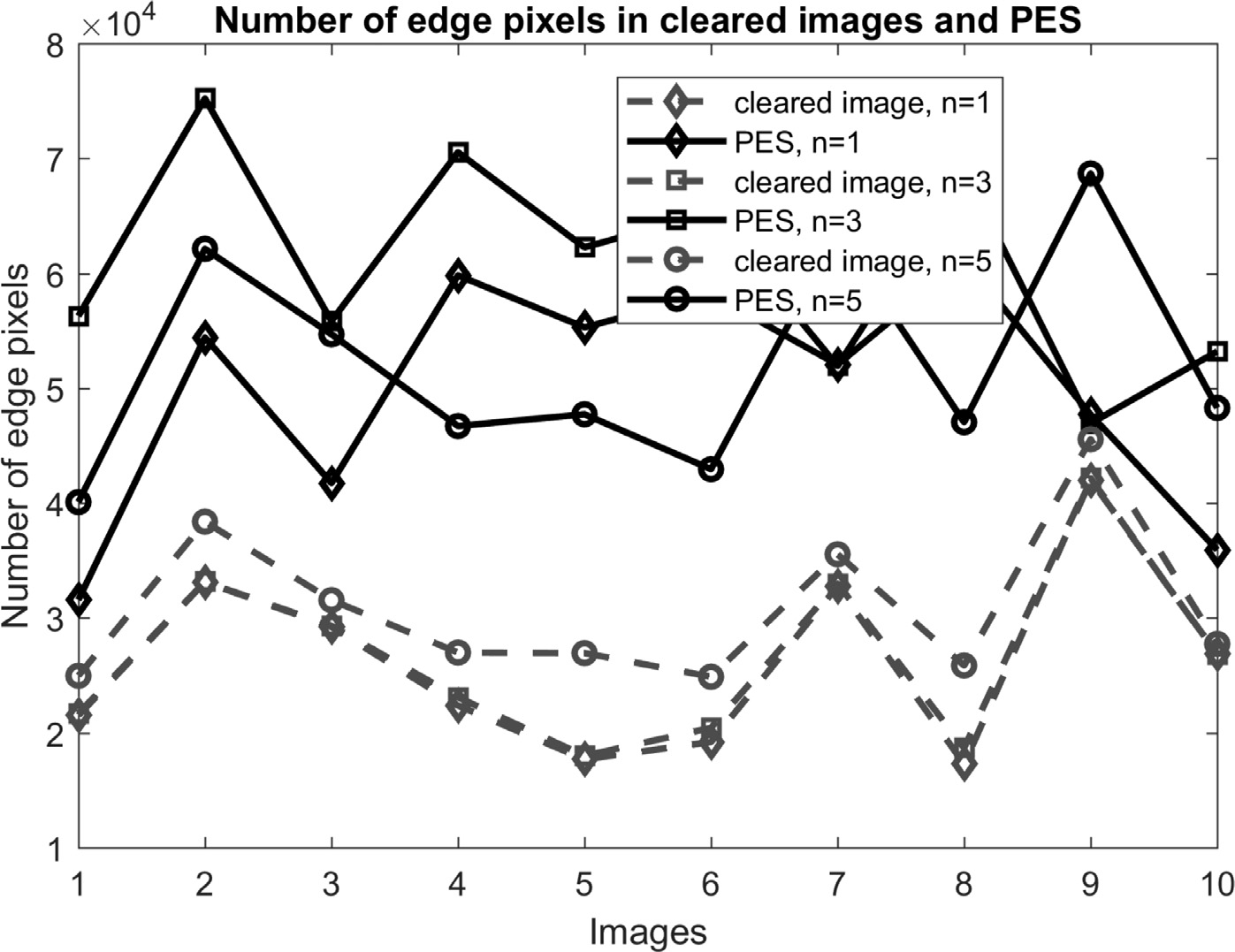
this table also explains that the proposed scheme offers much more pay-

loads than the alternatives. That the proposed scheme considerably en- hances the embedding capacity is therefore validated.

When a scheme implants its secrets in an image, obviously, it de-

that were implanted in an image per one dB of *𝑃 𝑆𝑁𝑅* loss. The mea- stroys the visual quality of that image. We measured the number of bits

surement is illustrated graphically in [Fig. 9](#_bookmark34). It is easy to see from the diagram that the proposed scheme implants a satisfactory amount of data as opposed to one dB PSNR loss, and it is higher than the compet- ing schemes. Again, the results of SSIM are figured out in [Table 5](#_bookmark51) for

**Fig. 6.** Number of edge pixels in image and prediction error space (PES). After clearing 1-bit LSB, we com- puted edge pixels that were detected by Canny. A sim- ilar approach was done for 3-bit and 5-bit LSB-cleared images. These are represented in the figure by solid lines and diamond, square, and circle markers, respec- tively. The same was done by applying our predictor and then employing Canny in PES. These results are presented by dash lines and diamond, square, and cir- cle markers, respectively. The figure states that Canny well performs in all PES-based experiments.

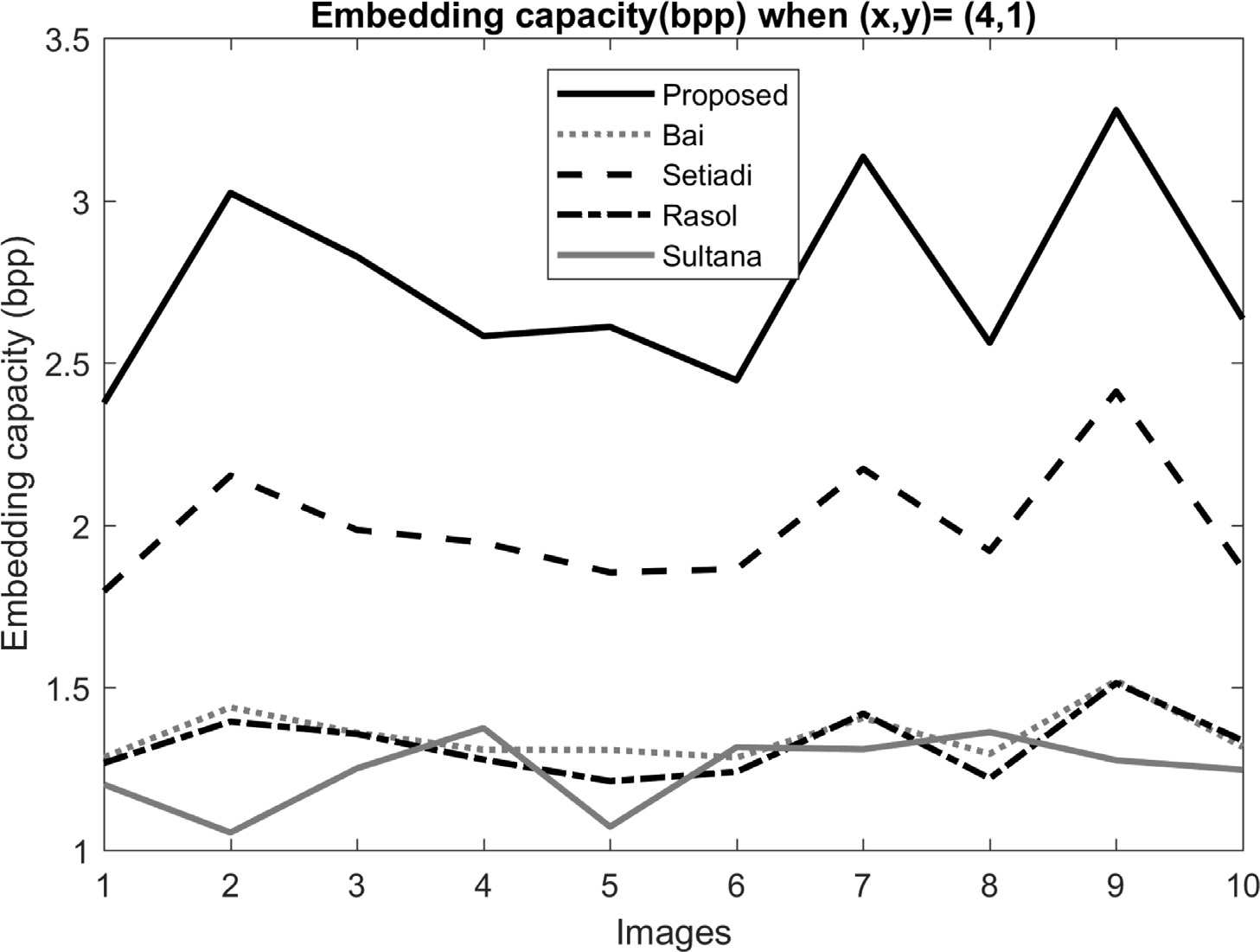
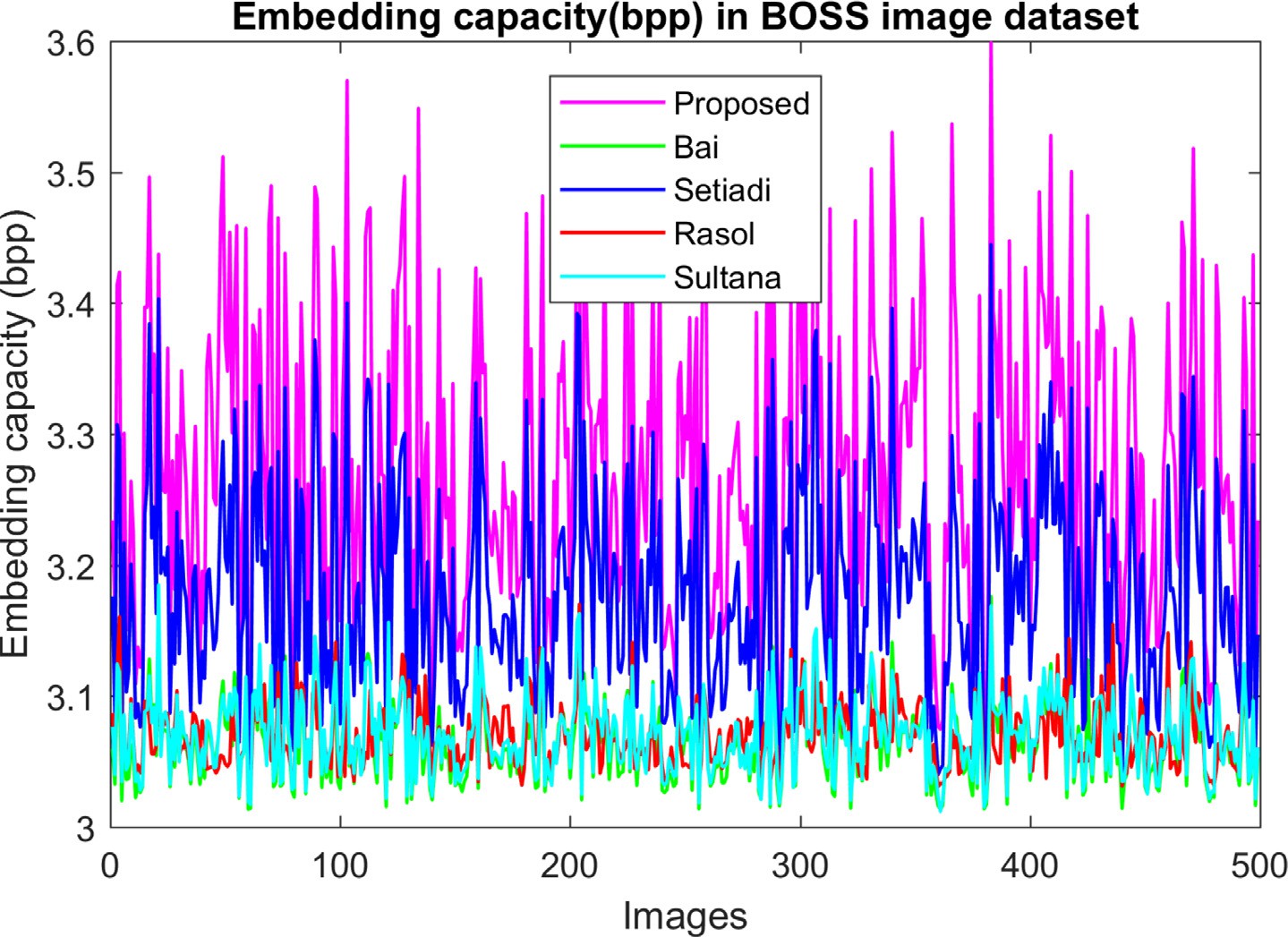
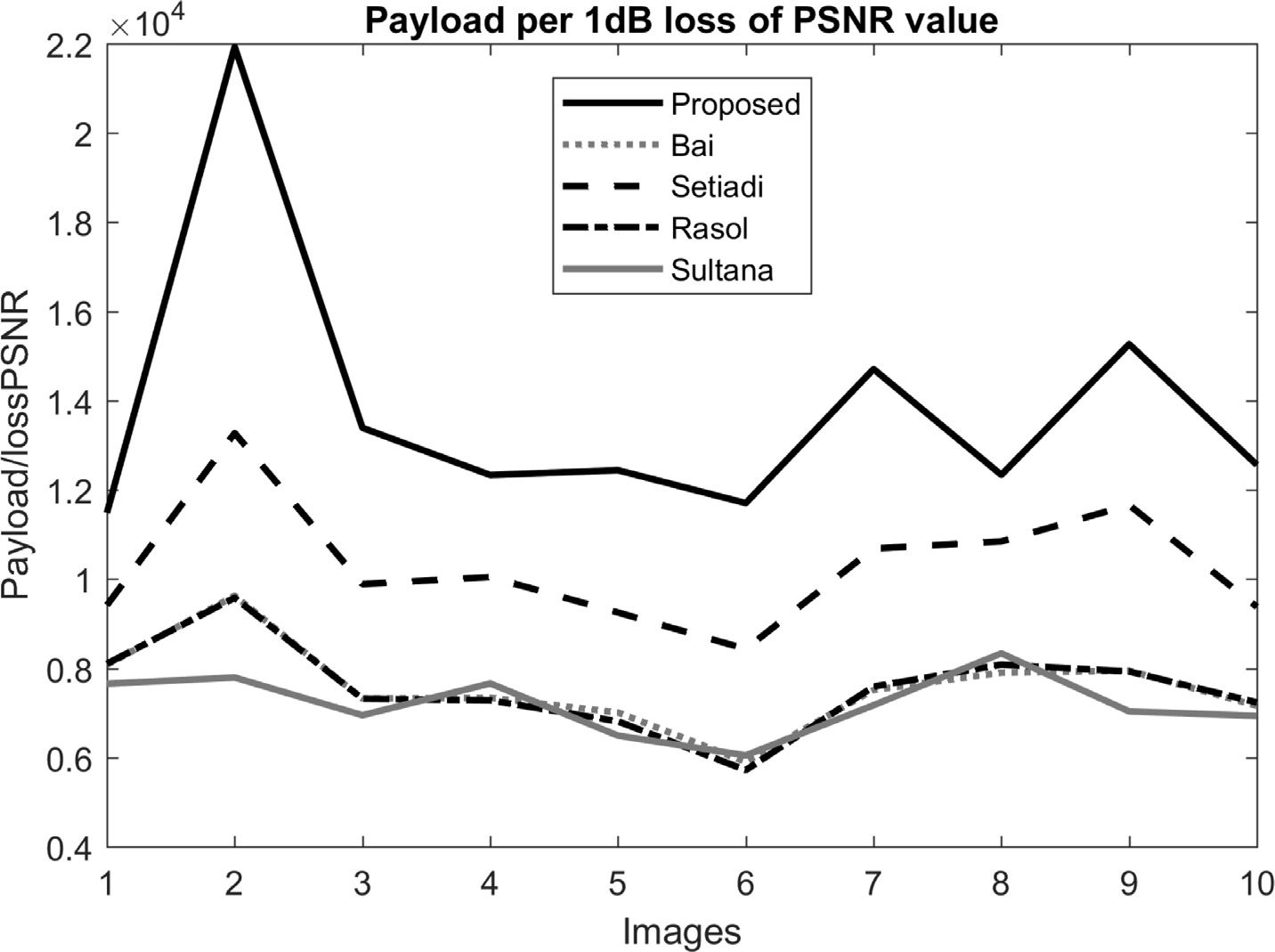
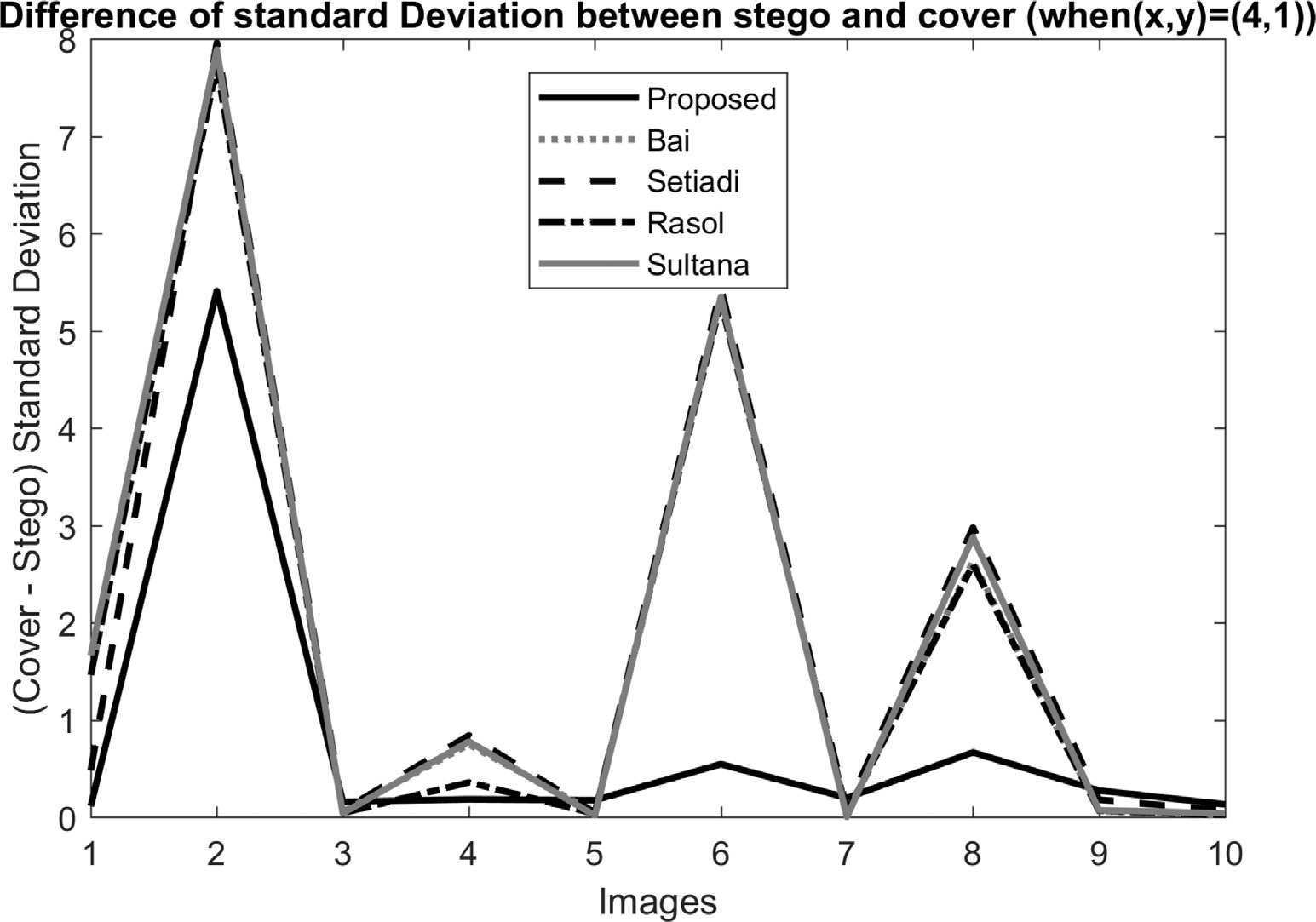
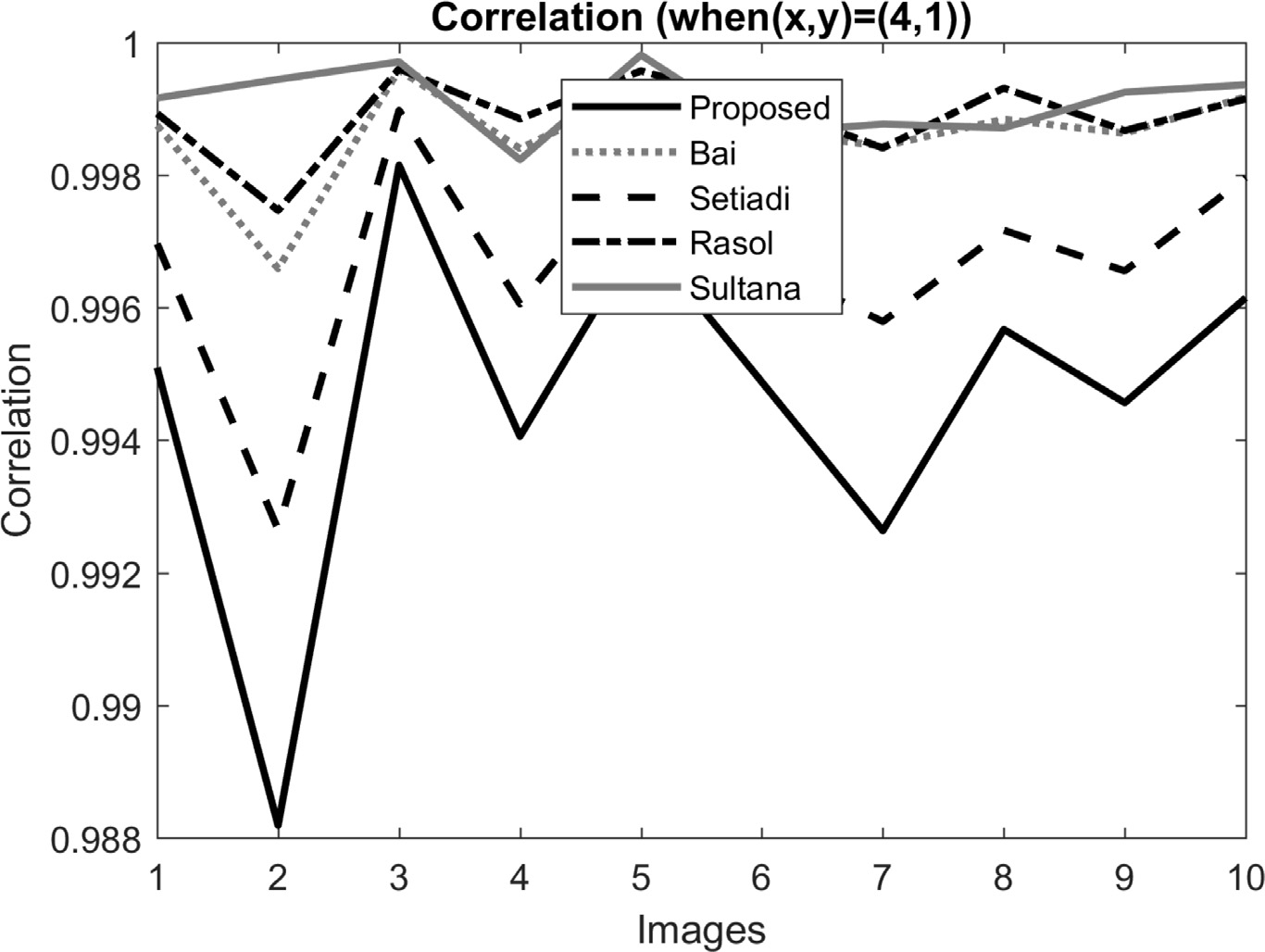
ure is generated by embedding at (x,y)=(4,1). The **Fig. 7.** Comparison of embedding rate. This fig-

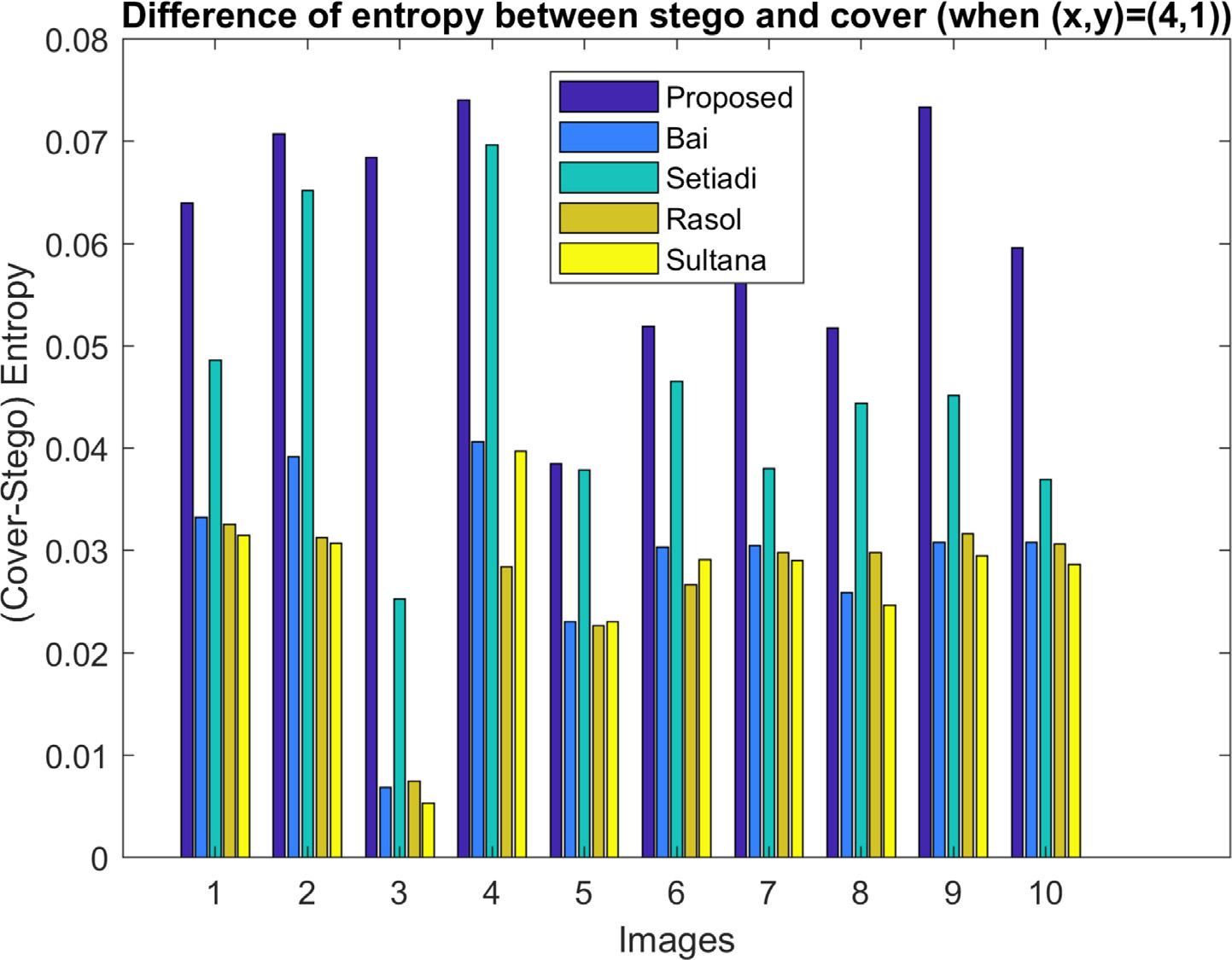
figure states that the proposed scheme yields em- bedding capacity at a dominating rate which is about 50% higher than Setiadi, 70% higher than Rasol et al., 80% higher than Bai at al. and 90% higher than Sultana et al.

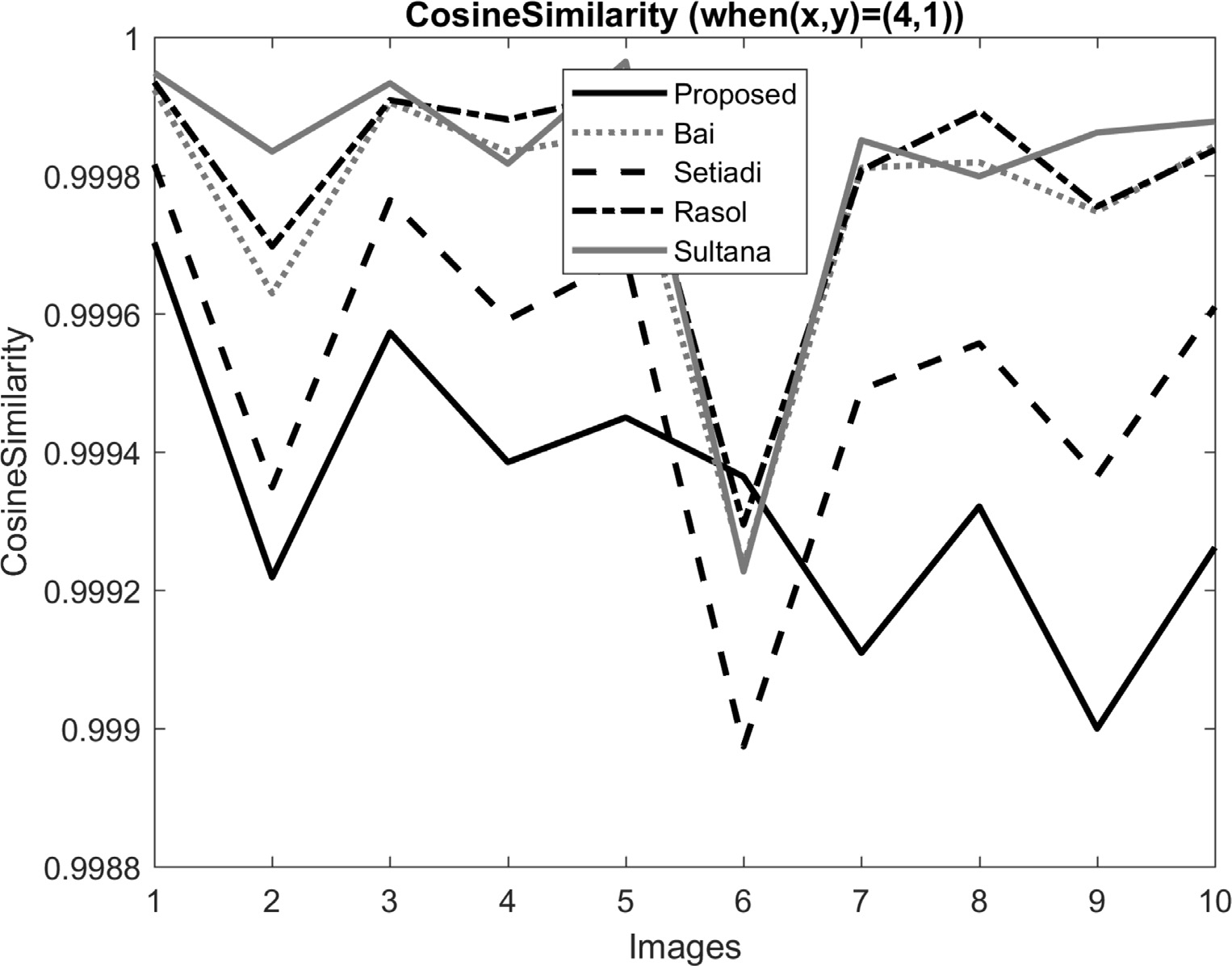
**Fig. 8.** Performance comparison of the proposed scheme with Bai, Setiadi, Sultana, and Rasol in terms of capacity in the BOSS image dataset. That experiments were done to justify the proposed scheme in a large image dataset. The figure states that our proposal is unbeaten in producing a larger embedding capacity.

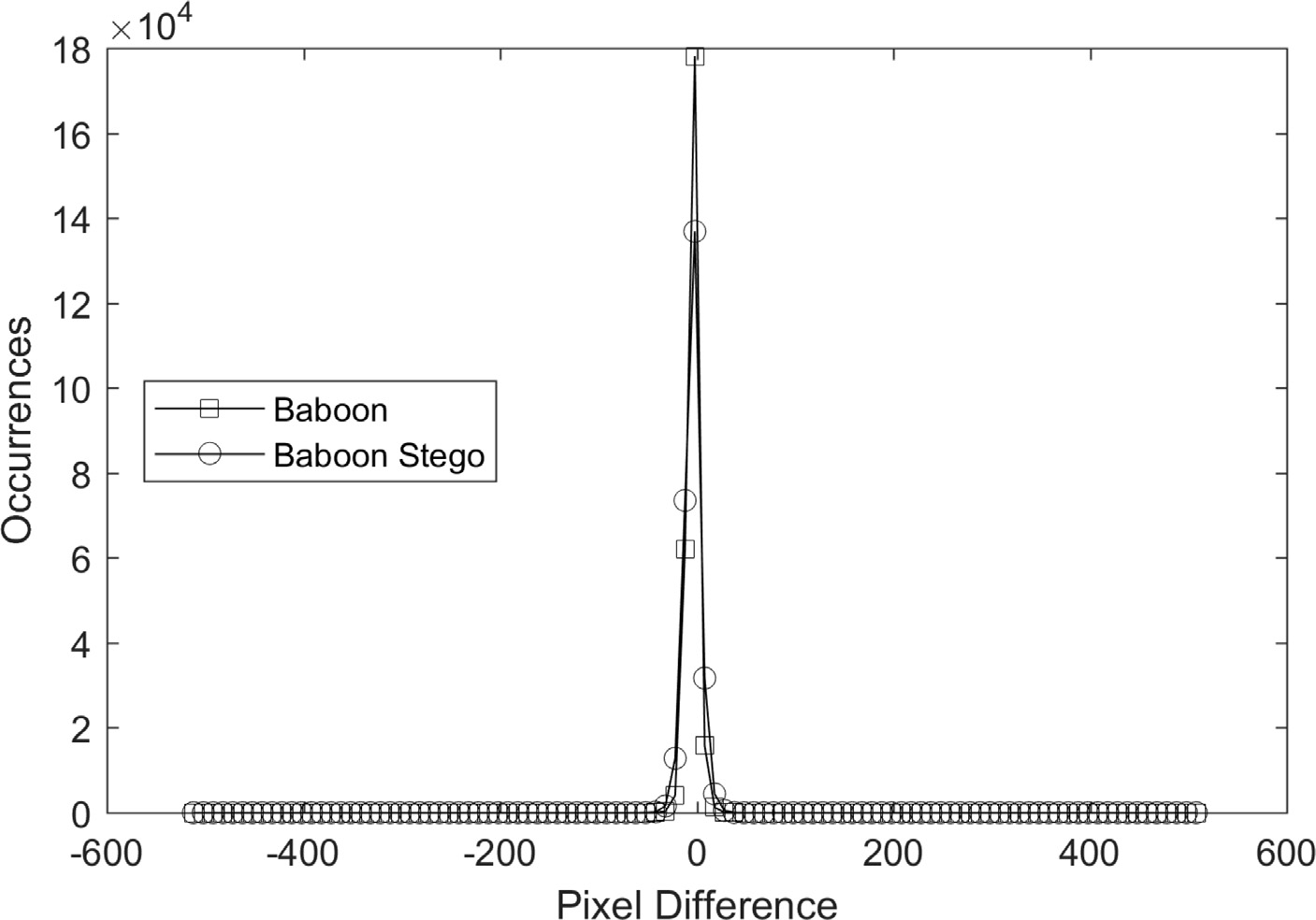
**Fig. 9.** Number of embedded bits per 1dB loss of PSNR. Regarding that per 1dB loss of PSNR, the pro- posed scheme noticeably dominates the other com- peting schemes.

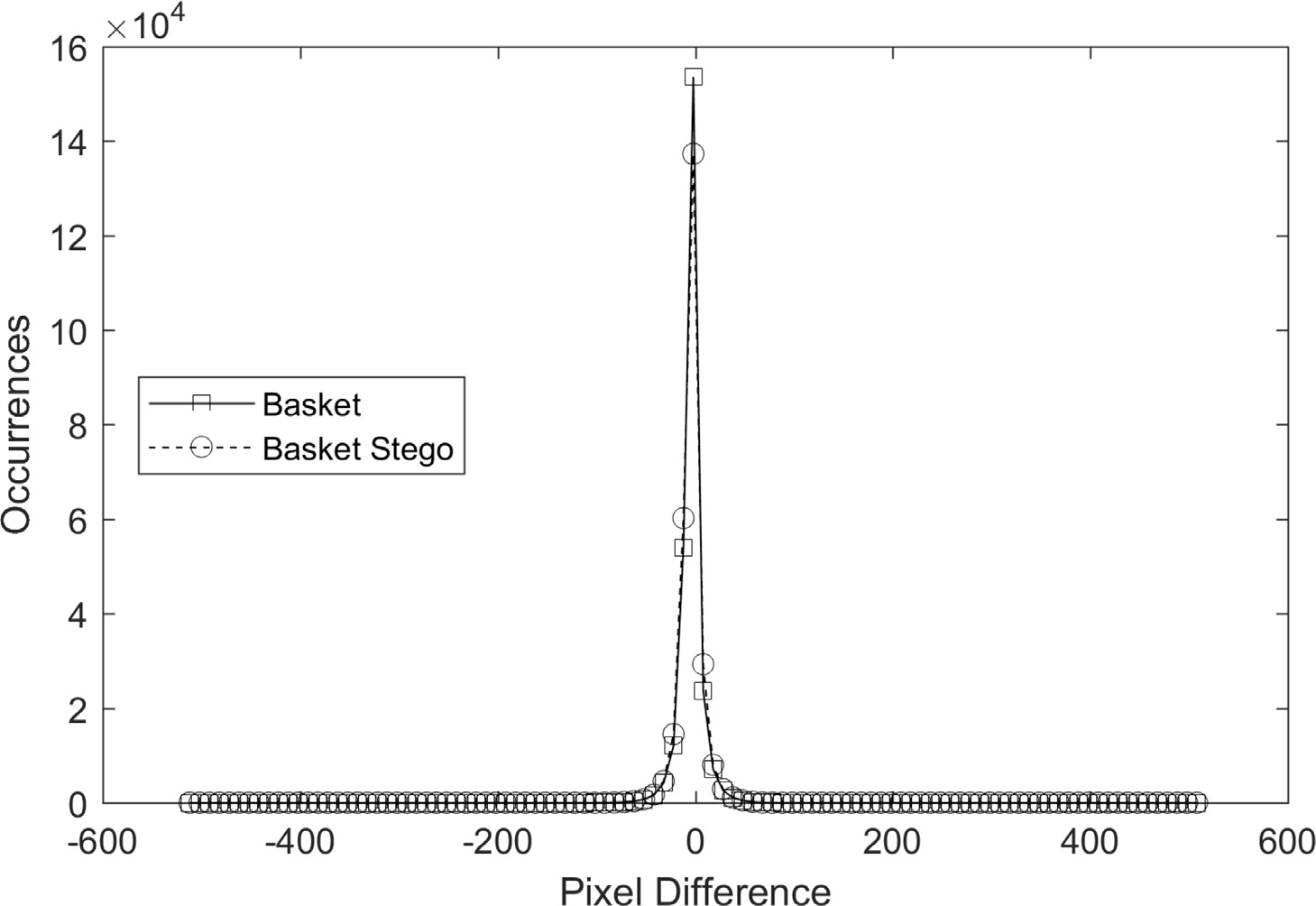
**Fig. 10.** Difference of standard Deviations of cover and stego images. The figure states that the differences are very small, close to each other. The proposed method is closer to zero.

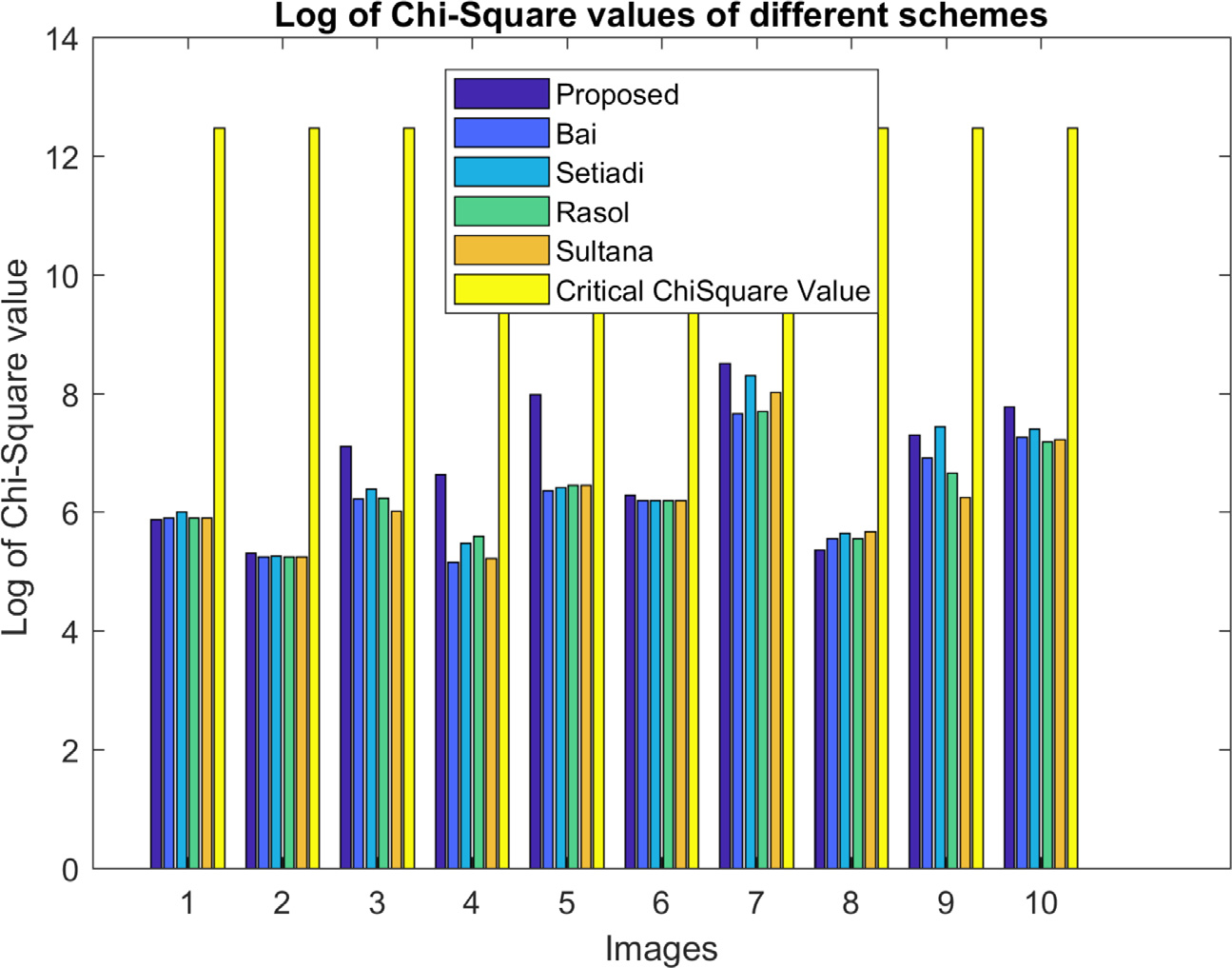
**Fig. 11.** Correlation coeﬃcients. All the schemes present higher correlations.

**Fig. 12.** Difference of entropy between cover and stego. The differences are very small and insignifi- cant to mention.

**Fig. 13.** Cosine similarity values. Measured cosine similarity values are very high in all the images.

**Fig. 14.** Pixel Difference Histogram of Baboon.

**Fig. 15.** Pixel Difference Histogram of Basket.

**Fig. 16.** Comparison of chi-square values among the schemes in the logarithmic scale.

**Table 4**

Execution time per image (in seconds).

parameters. PSNR is measured by [Eq. (22)](#_bookmark43).

*𝑃 𝑆𝑁𝑅* = 10 log

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image | Methods |  |  |  |  |
| 512 × 512 | Proposed | Bai | Setiadi | Sultana | Rasol |
| F16 | 14.40 | 13.37 | 13.90 | 13.34 | 13.52 |
| babon | 14.92 | 13.43 | 14.26 | 13.15 | 13.64 |
| basket | 14.75 | 13.42 | 14.03 | 13.46 | 13.53 |
| boat | 14.53 | 13.69 | 13.95 | 13.62 | 13.45 |
| brbra | 14.39 | 13.31 | 13.92 | 13.08 | 13.35 |
| lena | 14.36 | 13.30 | 13.91 | 13.56 | 13.39 |
| livingroom | 15.14 | 13.49 | 14.18 | 13.45 | 13.58 |
| pepper | 14.41 | 13.33 | 14.00 | 13.57 | 13.41 |
| walkbridge | 15.29 | 13.64 | 14.49 | 13.51 | 13.78 |
| wheel | 14.54 | 13.24 | 13.94 | 13.57 | 13.56 |

Where,

2552

10 *𝑀𝑆𝐸*

(22)

*𝑤 ℎ*

all the stated combinations of (*𝑥, 𝑦*). Depending on the distortion rate SSIM value ranges from 0 to 1. A higher value of SSIM signifies better originality of the image. The table confirms that the SSIM values of all

*𝑀𝑆𝐸* = 1 ∑ ∑(*𝑆* − *𝐶𝑖, 𝑗*)2;

here, *𝑆* is the stego image and *𝐶* is the original cover image.

*𝑖*=1 *𝑗*=1

*ℎ* ∗ *𝑤 𝑖,𝑗*

We expect to have our *𝑃 𝑆𝑁𝑅* value at 100dB. That *𝑃 𝑆𝑁𝑅* is, in-

age is a tempered image. Hence, the value of *𝑃 𝑆𝑁𝑅* will deteriorate. deed, achievable for two identical images only. However, the stego im- The amount of loss in *𝑃 𝑆𝑁𝑅* value, let *𝑙𝑜𝑠𝑠𝑃 𝑆𝑁𝑅*, is measured by [(23)](#_bookmark44).

*𝑙𝑜𝑠𝑠𝑃 𝑆𝑁𝑅* = (100 − *𝑃 𝑆𝑁𝑅*)*𝑑𝐵* (23)

Again, SSIM is calculated by [Eq. (24)](#_bookmark45).

(2*𝜇 𝜇* + *𝐶* )(2*𝜎* + *𝐶* )

the schemes are both high and very close to each other. Though these

*𝑆𝑆𝐼𝑀* =

*𝑐 𝑠*

1 *𝑐𝑠* 2

(24)

(*𝜇*2 + *𝜇*2 + *𝐶*1)(*𝜎*2 + *𝜎*2 + *𝐶*2)

schemes randomly dictate one another, there is nothing significant to

*𝑐 𝑠*

*𝑐 𝑠*

mention about the level of dictation. Thus, we can conclude that the proposed schemes do not destroy the quality of the image at any signif- icant level, and are even higher than the others.

* 1. *Mathematical representation of feature values*

the number of edge and non-edge pixels in a cover image be *𝑒𝑝𝑇* and Payload is the total number of implanted bits in the cover image. Let

*𝑛𝑝𝑇* , respectively. Then the maximum achievable payload *𝑃 𝐿* is defined

by [Eq. (20)](#_bookmark48).

*𝑃 𝐿* = *𝑒𝑝𝑇* × *𝑥* + *𝑛𝑝𝑇* × *𝑦* (20)

Sometimes, capacity is also measured to have a closer look at the per- formance of a scheme. Capacity means the number of implanted bits per

pixel. Embedding capacity, *𝐸𝐶*, is measured by [Eq. (21)](#_bookmark49)

In [Eq. (24)](#_bookmark45), *𝜇𝑐* and *𝜎𝑐* are the mean and variance of pixel values in the cover image. Likewise cover, *𝜇𝑠* and *𝜎𝑠* are the same for the stego image. Again, *𝜎𝑐𝑠* is the co-variance between the cover and stego image. *𝐶*1 and

*𝐶*2 are two constants. In experiment, we set *𝐶*1 = 0*.*0001 and *𝐶*2 = 0*.*0009

There are many methods of analyzing the security of a scheme. Very

simple but common ones are entropy measurement, analyzing correla- tion among the pixels, and checking the cosine similarity between the cover and stego image. The entropy is measured by Eq. [(25)](#_bookmark47).

∑

*𝐻* = − *𝑝𝑘* log2(*𝑝𝑘*) (25)

*𝑘*

where, *𝑝𝑘* is the probability associated with gray value *𝑘* and 1 ≤ *𝑘* ≤ 255.

In our experiment, we used the population correlation coeﬃcient.

Populations were measured from the pixel histogram. Population corre- lation is defined by [Eq. (26)](#_bookmark50).

*𝐸𝐶* = *𝑝*

*ℎ* × *𝑤*

Where *ℎ* and *𝑤* are the image height and width.

(21)

*𝜌* = *𝜎𝑐𝑠*

*𝑐𝑠 𝜎𝑐 𝜎𝑠*

(26)

While hiding data, maintaining image quality is a challenging issue. PSNR and SSIM are two commonly used image distortion measurement

where *𝜎𝑐* and *𝜎𝑠* are population standard deviations in cover *𝐶* and stego

*𝑆*. Again, *𝜎𝑐𝑠* is the co-variance between the cover and stego image.

*H. Sultana, A.H.M. Kamal, T.S. Apon et al.*

*Cyber Security and Applications 2 (2024) 100028*

16

**Table 5**

Comparison of the payload of the proposed scheme with its competing methods.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | Methods |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | (x,y)=(2,1) |  |  |  |  | (x,y)=(3,1) |  |  |  |  | (x,y)=(3,2) |  |  |  |  |
|  | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol |
| F16.jpg | 382,405 | 287,110 | 331,769 | 279,784 | 285,560 | 502,666 | 312,076 | 401,394 | 297,424 | 308,976 | 644,549 | 549,254 | 593,913 | 541,928 | 547,704 |
| babon.jpg | 438,942 | 300,527 | 362,871 | 266,862 | 296,710 | 615,740 | 338,910 | 463,598 | 271,580 | 331,276 | 701,086 | 562,671 | 625,015 | 529,006 | 558,854 |
| basket.jpg | 421,823 | 293,704 | 348,285 | 284,119 | 293,367 | 581,502 | 325,264 | 434,426 | 306,094 | 324,590 | 683,967 | 555,848 | 610,429 | 546,263 | 555,511 |
| boat.jpg | 400,418 | 289,135 | 344,871 | 294,957 | 286,470 | 538,692 | 316,126 | 427,598 | 327,770 | 310,796 | 662,562 | 551,279 | 607,015 | 557,101 | 548,614 |
| brbra.jpg | 402,900 | 289,085 | 336,837 | 268,416 | 280,665 | 543,656 | 316,026 | 411,530 | 274,688 | 299,186 | 665,044 | 551,229 | 598,981 | 530,560 | 542,809 |
| lena.jpg | 388,541 | 287,028 | 337,748 | 289,808 | 283,176 | 514,938 | 311,912 | 413,352 | 317,472 | 304,208 | 650,685 | 549,172 | 599,892 | 551,952 | 545,320 |
| livingroom.jpg | 448,662 | 297,687 | 364,715 | 289,262 | 298,866 | 635,180 | 333,230 | 467,286 | 316,380 | 335,588 | 710,806 | 559,831 | 626,859 | 551,406 | 561,010 |
| pepper.jpg | 398,623 | 288,004 | 342,641 | 293,868 | 281,403 | 535,102 | 313,864 | 423,138 | 325,592 | 300,662 | 660,767 | 550,148 | 604,785 | 556,012 | 543,547 |
| walkbridge.jpg | 461,213 | 307,707 | 385,532 | 286,277 | 307,067 | 660,282 | 353,270 | 508,920 | 310,410 | 351,990 | 723,357 | 569,851 | 647,676 | 548,421 | 569,211 |
| wheel.jpg | 405,059 | 289,889 | 337,832 | 283,729 | 291,563 | 547,974 | 317,634 | 413,520 | 305,314 | 320,982 | 667,203 | 552,033 | 599,976 | 545,873 | 553,707 |

**Table 6**

Comparison of the payload of the proposed scheme with its competing methods.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | Methods |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | (x,y)=(4,1) |  |  |  |  | (x,y)=(4,2) |  |  |  |  | (x,y)=(4,3) |  |  |  |  |
|  | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol |
| F16.jpg | 622,927 | 337,042 | 471,019 | 315,064 | 332,392 | 764,810 | 574,220 | 663,538 | 559,568 | 571,120 | 906,693 | 811,398 | 856,057 | 804,072 | 809,848 |
| babon.jpg | 792,538 | 377,293 | 564,325 | 276,298 | 365,842 | 877,884 | 601,054 | 725,742 | 533,724 | 593,420 | 963,230 | 824,815 | 887,159 | 791,150 | 820,998 |
| basket.jpg | 741,181 | 356,824 | 520,567 | 328,069 | 355,813 | 843,646 | 587,408 | 696,570 | 568,238 | 586,734 | 946,111 | 817,992 | 872,573 | 808,407 | 817,655 |
| boat.jpg | 676,966 | 343,117 | 510,325 | 360,583 | 335,122 | 800,836 | 578,270 | 689,742 | 589,914 | 572,940 | 924,706 | 813,423 | 869,159 | 819,245 | 810,758 |
| brbra.jpg | 684,412 | 342,967 | 486,223 | 280,960 | 317,707 | 805,800 | 578,170 | 673,674 | 536,832 | 561,330 | 927,188 | 813,373 | 861,125 | 792,704 | 804,953 |
| lena.jpg | 641,335 | 336,796 | 488,956 | 345,136 | 325,240 | 777,082 | 574,056 | 675,496 | 579,616 | 566,352 | 912,829 | 811,316 | 862,036 | 814,096 | 807,464 |
| livingroom.jpg | 821,698 | 368,773 | 569,857 | 343,498 | 372,310 | 897,324 | 595,374 | 729,430 | 578,524 | 597,732 | 972,950 | 821,975 | 889,003 | 813,550 | 823,154 |
| pepper.jpg | 671,581 | 339,724 | 503,635 | 357,316 | 319,921 | 797,246 | 576,008 | 685,282 | 587,736 | 562,806 | 922,911 | 812,292 | 866,929 | 818,156 | 805,691 |
| walkbridge.jpg | 859,351 | 398,833 | 632,308 | 334,543 | 396,913 | 922,426 | 615,414 | 771,064 | 572,554 | 614,134 | 985,501 | 831,995 | 909,820 | 810,565 | 831,355 |
| wheel.jpg | 690,889 | 345,379 | 489,208 | 326,899 | 350,401 | 810,118 | 579,778 | 675,664 | 567,458 | 583,126 | 929,347 | 814,177 | 862,120 | 808,017 | 815,851 |

**Table 7**

Comparison of SSIM results of the proposed scheme with other competing schemes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | Methods |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | (x,y)=(2,1) |  |  |  |  | (x,y)=(3,1) |  |  |  |  | (x,y)=(3,2) |  |  |  |  |
|  | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol |
| F16.jpg | 0.989 | 0.993 | 0.992 | 0.994 | 0.993 | 0.973 | 0.990 | 0.984 | 0.993 | 0.992 | 0.960 | 0.973 | 0.969 | 0.976 | 0.974 |
| babon.jpg | 0.962 | 0.972 | 0.969 | 0.974 | 0.972 | 0.927 | 0.962 | 0.948 | 0.973 | 0.967 | 0.924 | 0.949 | 0.940 | 0.956 | 0.953 |
| basket.jpg | 0.994 | 0.997 | 0.996 | 0.997 | 0.997 | 0.980 | 0.995 | 0.991 | 0.997 | 0.995 | 0.974 | 0.986 | 0.983 | 0.988 | 0.986 |
| boat.jpg | 0.988 | 0.993 | 0.991 | 0.993 | 0.994 | 0.965 | 0.987 | 0.977 | 0.986 | 0.991 | 0.955 | 0.972 | 0.964 | 0.971 | 0.975 |
| brbra.jpg | 0.989 | 0.994 | 0.992 | 0.995 | 0.995 | 0.965 | 0.989 | 0.980 | 0.995 | 0.993 | 0.957 | 0.975 | 0.968 | 0.980 | 0.978 |
| lena.jpg | 0.982 | 0.987 | 0.984 | 0.987 | 0.988 | 0.960 | 0.980 | 0.969 | 0.980 | 0.984 | 0.946 | 0.962 | 0.954 | 0.962 | 0.966 |
| livingroom.jpg | 0.993 | 0.997 | 0.996 | 0.997 | 0.997 | 0.974 | 0.995 | 0.990 | 0.997 | 0.995 | 0.972 | 0.988 | 0.984 | 0.989 | 0.988 |
| pepper.jpg | 0.984 | 0.990 | 0.988 | 0.990 | 0.991 | 0.957 | 0.983 | 0.971 | 0.982 | 0.989 | 0.948 | 0.966 | 0.955 | 0.966 | 0.971 |
| walkbridge.jpg | 0.995 | 0.998 | 0.997 | 0.998 | 0.998 | 0.981 | 0.995 | 0.991 | 0.998 | 0.996 | 0.979 | 0.991 | 0.987 | 0.992 | 0.991 |
| wheel.jpg | 0.993 | 0.996 | 0.995 | 0.996 | 0.996 | 0.980 | 0.994 | 0.992 | 0.996 | 0.995 | 0.970 | 0.982 | 0.979 | 0.983 | 0.982 |

**Table 8**

Comparison of SSIM results of the proposed scheme with other competing schemes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | Methods |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | (x,y)=(4,1) |  |  |  |  | (x,y)=(4,2) |  |  |  |  | (x,y)=(4,3) |  |  |  |  |
|  | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol | Proposed | Bai | Setiadi | Sultana | Rasol |
| F16.jpg | 0.925 | 0.975 | 0.955 | 0.992 | 0.984 | 0.914 | 0.959 | 0.942 | 0.975 | 0.968 | 0.879 | 0.911 | 0.895 | 0.922 | 0.917 |
| babon.jpg | 0.832 | 0.924 | 0.878 | 0.971 | 0.946 | 0.826 | 0.911 | 0.874 | 0.958 | 0.934 | 0.804 | 0.870 | 0.838 | 0.899 | 0.883 |
| basket.jpg | 0.933 | 0.987 | 0.972 | 0.995 | 0.989 | 0.928 | 0.978 | 0.964 | 0.986 | 0.981 | 0.909 | 0.947 | 0.937 | 0.953 | 0.950 |
| boat.jpg | 0.892 | 0.964 | 0.929 | 0.962 | 0.980 | 0.882 | 0.949 | 0.919 | 0.947 | 0.965 | 0.852 | 0.903 | 0.879 | 0.902 | 0.913 |
| brbra.jpg | 0.891 | 0.966 | 0.935 | 0.995 | 0.986 | 0.883 | 0.952 | 0.925 | 0.979 | 0.972 | 0.856 | 0.907 | 0.886 | 0.927 | 0.921 |
| lena.jpg | 0.890 | 0.953 | 0.915 | 0.954 | 0.973 | 0.879 | 0.936 | 0.901 | 0.937 | 0.955 | 0.847 | 0.886 | 0.860 | 0.886 | 0.899 |
| livingroom.jpg | 0.913 | 0.986 | 0.967 | 0.994 | 0.988 | 0.910 | 0.979 | 0.961 | 0.987 | 0.981 | 0.901 | 0.953 | 0.939 | 0.959 | 0.955 |
| pepper.jpg | 0.877 | 0.954 | 0.913 | 0.950 | 0.979 | 0.868 | 0.937 | 0.901 | 0.935 | 0.962 | 0.828 | 0.887 | 0.859 | 0.885 | 0.900 |
| walkbridge.jpg | 0.931 | 0.986 | 0.968 | 0.996 | 0.989 | 0.930 | 0.982 | 0.965 | 0.991 | 0.984 | 0.925 | 0.965 | 0.951 | 0.971 | 0.966 |
| wheel.jpg | 0.937 | 0.989 | 0.978 | 0.994 | 0.990 | 0.928 | 0.976 | 0.966 | 0.981 | 0.978 | 0.902 | 0.938 | 0.930 | 0.942 | 0.939 |

Equation [(27)](#_bookmark54) gives us the cosine similarity values.

*𝑓* (*𝐶, 𝑆*) = *𝑐𝑜𝑠*(*𝜃*)

statistics, we measured correlation coeﬃcients *𝜌𝑆 𝐶* between the cover than the others and it is closer to zero. To verify further with similar

*𝐶𝑜𝑠𝑆𝑖𝑚*

∑*ℎ* ∑*𝑤*

and stego image. *𝜌𝑆 𝐶 >* 0 means a positive correlation between cover

= *ℎ*

√∑

∑

*𝑖*=1 *𝑗*=1 *𝐶*(*𝑖,𝑗*)*𝑆*(*𝑖,𝑗*)

*𝑖*=1

*𝑤*

*𝑗*=1

*𝐶*(*𝑖,𝑗*)√∑*ℎ*

*𝑤*

*𝑗*=1

∑

*𝑆*(*𝑖,𝑗*)

and stego image and *𝜌𝑆 𝐶* signifies a perfect relationship when it reaches

to 1. Similarly, a negative value of *𝜌𝑆 𝐶* indicates a negative relationship.

(27)

where *𝐶* and *𝑆* are cover and stego images.

*𝑖*=1

When they are the same, the function *𝑓𝐶𝑜𝑠𝑆𝑖𝑚* provides the highest value, which is 1. *𝑓𝐶𝑜𝑠𝑆𝑖𝑚*’s computed value depends on the tempering effect in image *𝑆*. The more tempered in *𝑆*, the smaller the value of

*𝑓𝐶𝑜𝑠𝑆𝑖𝑚*.

* 1. *Analyzing the complexity*

We compare the time complexity of different schemes by analyzing their execution time and compared with existing studies and result is shown in [Table 4](#_bookmark42). In terms of completion time author Bai’s study outper- formed every other study. On average author’s proposed scheme require about 13.42 s [[24]](#_bookmark70). Similar to Bai’s study, Sultana and Rasol’s proposed framework requires 13.43 and 13.52 s [[32,34]](#_bookmark78). Setiadi’s study on the other hand requires 14.06 s [[25]](#_bookmark71). On average, our suggested method consumes 14.67 s per image. Even though it ends up taking a little bit longer, it exceeds all previous research and yields significantly superior outcomes as a consequence. Thus this difference of 0.5–1 s can be over- looked.

* 1. *Security analysis*

We statistically analyzed our scheme to check its robustness against various attacks. We first measured the standard deviation of pixel values from their mean, separately, in cover and stego images. Let the standard

deviation in cover and stego image is *𝜎𝑐* and *𝜎𝑠*. Next, we calculated

*𝜌𝑆 𝐶* = 0 stands for no relationship between two images. Results of *𝜌𝑆 𝐶*

are depicted in [Fig. 11](#_bookmark36). Though the proposed method shows a lower

correlation value, its difference from others is insignificance. Rather, as

with others, it represents a higher correlation between cover and stego.

We measured the entropy values *𝐻* as well. We computed *𝐻* both

in cover and stego images. Next, we calculated their differences. That

difference value is zero for two identical images. Results are plotted in [Fig. 12](#_bookmark37). The figure shows that none of the results are greater than 0.08, i.e., these are very small and close to zero. We also computed cosine similarities between cover and stego images. That value is 1 for two identical images and 0 for two fully mismatched images. The results are demonstrated in [Fig. 13](#_bookmark38). The figure illustrates that all the values are greater than 0.999, which is very high. Besides, the values in all the schemes are very close to each other, where, the maximum variation is o.0008, i.e., 0.9998-0.999.

Thus, it can be deduced from the results of these experiments that our method is strong enough to protect against attacks on implanted data.

We used statistical tools of pixel difference histogram (PDH) to iden- tify the stego images. The PDH of the original images and corresponding stego images are shown in [Figs. 14](#_bookmark39) and [15](#_bookmark40).

We also used another statistical tool of Chi-Square test for determin- ing the difference between stego and cover images. This test is used to find out whether a difference between two images is due to chance or a relationship between them. The experimental result of the Chi-square test is shown in [Fig. 16](#_bookmark41).

their difference by *𝜎𝑑* = *𝜎𝑐* − *𝜎𝑠*. Ideally, *𝜎𝑑* should be zero for a non-

tempered image. That *𝜎𝑑* is drawn in [Fig. 10](#_bookmark35) against different images.

The results show that the proposed scheme produces smaller differences

*𝜒𝐷𝐹* 2 =

(*𝑆𝑖* − *𝐶𝑖*)2

*𝐶𝑖*

∑

(28)

where *𝐷𝐹* is the degrees of freedom, *𝐶* and *𝑆* are cover and stego im- ages. *𝐷𝐹* is calculated by ([Fig. 16](#_bookmark41))

*𝐷𝐹* = (*ℎ* − 1) ∗ (*𝑤* − 1) (29)

Here *ℎ* and *𝑤* are the height and width of the cover and stego images

([Tables 6–8](#_bookmark51)).

# Discussion

Using this research, it can secure both messages and communicating parties. No intruder can able to receive some beneficial information from the sending file during transmission. Besides Corporations government and law enforcement agencies can connect privately and communicate secretly.

Although this model has many advantages it also has some limi- tations. This model cannot work without edge detectors. This model consumes time when applying predictor and it is not a big matter. If a steganography approach generates someone to suspect the carrier medium, thus the model has been unsuccessful.

# Conclusion

This research first time proposes the idea of grouping image pixels as edge and non-edge by applying an edge detector in its prediction er- ror space. The research first generates a prediction error space from a cover image and then applies a desired predictor in that created error space. The locations of detected edge errors are mapped in the cover image to classify edge and non-edge pixels. That strategy, significantly, increases the number of edge pixels. As the edge detection-based embed- ding schemes implant more bits in edge pixels than non-edge pixels, the same rules then well perform in the proposed method. The experimental results deduce that the scheme does not compromise the visual or struc- tural quality of the cover image more than the other competing schemes. Moreover, statistical analyses exhibit that the proposed scheme demon- strates stronger security against attacks. In the future, we hope to work on making the scheme reversible. At the same time, we would like to repeatedly embed an image with a back-and-forward strategy. In the back-and-forward strategy, we will increase the present values of pixels in the first cycle of data embedment and decrease the updated values in the second cycle of data embedment. Thus, we want to implant any length of the message in an image by managing its visual quality. In that case, managing the scheme’s reversibility is a pre-requisition. We also would like to work with other media such as audio, video, and text. We will apply this model for forensic or other security purposes.

# Availability of data and materials

We used data from “The Bank of Standardized Stimuli (BOSS), a New Set of 480 Normative Photos of Objects to Be Used as Visual Stimuli in Cognitive Research, Mathieu B. Brodeur, Emmanuelle Dionne-Dostie, Tina Montreuil, Martin Lepage” and various reliable sources (Internet).

# CRediT authorship contribution statement

**Habiba Sultana:** Conceptualization, Methodology, Soft- ware. **A.H.M. Kamal:** Conceptualization, Methodology, Software. **Tasnim Sakib Apon:** Data curation, Writing – original draft.

# Acknowledgments

The first author is a fellow of the ICT division of the Ministry of Post, Telecommunication, and Information Technology of the Government of Bangladesh. Therefore, we want to devote a thank to the concerned min- istry, and at the same time, we would like to acknowledge that support.

# References

1. [H. Al-Dmour, A. Al-Ani, A steganography embedding method based on edge identi- fication and XOR coding, Expert Syst. Appl. 46 (2016) 293–306.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0001)
2. [H. Sultana, A study on steganography and steganalysis, Int. J. Sci. Eng. Res. 9 (12) (2018).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0002)
3. [H. Sultana, A.H.M. Kamal, An edge detection based reversible data hiding scheme, in: 2022 IEEE Delhi Section Conference (DELCON), IEEE, 2022.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0003)
4. [P.W. Adi, F.Z. Rahmanti, N.A. Abu, High quality image steganography on integer harr wavelet transform using modulus function, in: International Conference on Sci- ence in Information Technology (ICSITech), IEEE, 2015.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0004)
5. [H. Sultana, A.H.M. Kamal, M.M. Islam, Enhancing the robustness of visual degrada- tion based HAM reversible data hiding, J. Comput. Sci. 12 (2) (2016) 88–97.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0005)
6. [A.H.M. Kamal, M.M. Islam, Enhancing embedding capacity and stego image quality by employing multi predictors, J. Inf. Secur. Appl. 32 (2017) 59–74.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0006)
7. [A.H.M. Kamal, M.M. Islam, Boosting up the data hiding rate through multi cycle embedment process, J. Vis. Commun. Image Represent. 40 (2016) 574–588.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0007)
8. [A.H.M. Kamal, M.M. Islam, Capacity improvement of reversible data hiding scheme through better prediction and double cycle embedding process, in: 2015 IEEE In- ternational Conference on Advanced Networks and Telecommuncations Systems (ANTS), IEEE, 2015.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0008)
9. [A.H.M. Kamal, M.M. Islam, An image distortion-based enhanced embedding scheme, Iran J. Comput. Sci. 1 (3) (2018) 175–186.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0009)
10. [A.H.M. Kamal, M.M. Islam, A prediction error based histogram association and map- ping technique for data embedment, J. Inf. Secur. Appl. 48 (2019) 102368.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0010)
11. [A.H.M. Kamal, M.M. Islam, Z. Islam, An embedding technique for smartcard-sup- ported e-healthcare services, Iran J. Comput. Sci. 3 (4) (2020) 195–205.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0011)
12. [A.H.M. Kamal, M.M. Islam, Enhancing the embedding payload by handling the affair of association and mapping of block pixels through prediction errors histogram, in: 2016 International Conference on Networking Systems and Security (NSysS), IEEE, 2016.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0012)
13. [W. Hong, Adaptive reversible data hiding method based on error energy control and histogram shifting, Opt. Commun. 285 (2012) 101–108.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0013)
14. [H. Yao, C. Qin, Z. Tang, Y. Tian, Improved dual-image reversible data hiding method using the selection strategy of shiftable pixels’ coordinates with minimum distortion, Signal Process. 135 (2016) 26–35.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0014)
15. [S. Yi, Y. Zhou, Binary-block embedding for reversible data hiding in encrypted im- ages, Signal Process. 133 (2017) 40–51.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0015)
16. [A.H.M. Kamal, M.M. Islam, Enhancing the performance of the data embedment pro- cess through encoding errors, Electronics (Basel) 5 (4) (2016) 79.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0016)
17. [A.H.M. Kamal, M.M. Islam, Facilitating and securing oﬄine e-medicine service through image steganography, Healthc. Technol. Lett. 1 (2) (2014) 74–79.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0017)
18. [M.D. Hasan, M.A.M. Amin, S.T. Mahdi, Steganalysis techniques and comparison of available softwares, Cyberspace (2020).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0018)
19. [C.F. Lee, W. C-Y, C. K-C, An eﬃcient reversible data hiding with reduplicated exploit- ing modification direction using image interpolation and edge detection, Multimed. Tools Appl. (2016).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0019)
20. [S. Kumar, A. Singh, M. Kumar, Information hiding with adaptive steganography based on novel fuzzy edge identification, Defence Technol. 15 (2) (2019) 162–169.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0020)
21. [S. Sun, A novel edge based image steganography with 2*𝑘* correction and Huffman](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0021)

[encoding, Inf. Process. Lett. (2015).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0021)

1. [C. Vanmathi, S. Prabu, Image steganography using fuzzy logic and chaotic for large payload and high imperceptibility, Int. J. Fuzzy Syst. (2017).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0022)
2. [G. Swain, Adaptive pixel value differencing steganography using both vertical and horizontal edges, Multimed. Tools Appl. 75 (21) (2016) 13541–13556.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0023)
3. [J. Bai, C.-C. Chang, T.-S. Nguyen, C. Zhu, Y. Liu, A high payload steganographic algorithm based on edge detection, Displays 46 (2017) 42–51.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0024)
4. D.R. Setiadi, Improved payload capacity in LSB image steganography uses dilated hybrid edge detection.
5. [E.J. Kusuma, O.R. Indriani, C.A. Sari, E.H. Rachmawanto, D.R.I.M. Setiadi, An im- perceptible LSB image hiding on edge region using DES encryption, in: 2017 Inter- national Conference on Innovative and Creative Information Technology (ICTTech), IEEE, 2018.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0025)
6. [S. Khan, N. Ahmad, M. Ismail, N. Minallah, T. Khan, A secure true edge based 4 least significant bits steganography, IEEE, 2015.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0026)
7. [W.J. Chen, C.C. Chang, T.H.N. Le, High payload steganography mechanism using hybrid edge detector, Expert Syst. Appl. 37 (2010).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0027)
8. [H.W. Tseng, H.S. Leng, High-paylaod block-based data hiding scheme using hybrid edge detector with minimal distortion, IET Image Process. 8 (2014).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0028)
9. [K. Gaurav, U. Ghanekar, Image steganography based on canny edge detection, dila- tion operator and hybrid coding, J. Inf. Secur. Appl. 41 (2018) 41–51.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0029)
10. [B. Vishnu, S.R. Sajeesh, L.V. Namboothiri, Enhanced image steganography with PVD and edge detection, in: 2020 Fourth International Conference on Computing Method- ologies and Communication (ICCMC), IEEE, 2020.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0030)
11. [H. Sultana, A.H.M. Kamal, Image steganography system based on hybrid edge de- tector, in: 2021 24th International Conference on Computer and Information Tech- nology (ICCIT), IEEE, 2021.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0031)
12. [D.R.I.M. Setiadi, J. Jumanto, An enhanced LSB-image steganography using the hy- brid canny-sobel edge detection, Cybern. Inf. Technol. 18 (2) (2018).](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0032)
13. [M.B. Brodeur, E. Dionne-Dostie, T. Montreuil, M. Lepage, The bank of standardized stimuli (BOSS), a new set of 480 normative photos of objects to be used as visual stimuli in cognitive research, PLoS ONE 5 (2010) e10773.](http://refhub.elsevier.com/S2772-9184(23)00016-4/sbref0033)