Chapter 3

Single Image Based Steganographic Scheme (SS)

Steganographic schemes in spatial domain are mainly designed to increase the embedding capacity of the stego image. The major drawback of this approach is to compromise with the visual quality of the stego image. So, designing a steganographic scheme in spatial domain with high embedding capacity without compromising the visual quality of the stego image is still a challenge. Here, steganographic schemes have been designed in spatial domain to increase the embedding capacity along with the good imperceptibility of the stego image.

Two steganographic schemes have been discussed here. Different approaches have been considered to design steganographic schemes using graph neighbourhood and PVD. In the first approach, a secured stenographic scheme (SSGN) has been developed using the concept of graph neighbourhood. The proposed technique employs a weighted matrix during embedding of the secret data. It helps to embed multiple bits in a pixel block by changing only a single pixel of the block. Moreover, the concept of graph neighbourhood helps to select the pixels in the same range for data embedding. This increases the visual quality of a stego image keeping the embedding capacity reasonably high. The effectiveness of the proposed scheme is judged using standard evaluation metrics like PSNR, SSIM, BER, and Q-Index with different threshold values. The average values of PSNR, SSIM, BER, and Q-Index are calculated to measure the strength of the proposed scheme. The PSNR of the proposed scheme is calculated after embedding maximum secret bits in the cover image.

In the second approach, a PVD based steganographic scheme (SSPVD) has been contemplated. PVD based steganographic schemes are not reversible. The reversibility has been accomplished in the proposed technique through image interpolation. Moreover, controlling the overflow and underflow situations is an overhead in some existing PVD based schemes. The overflow and underflow conditions are managed by changing the LSBs instead of adding values to the existing pixel values. This will ensure that the stated conditions never arise. The SSPVD scheme gains payload 2.22 bpp, that is better than some recently developed PVD based data hiding schemes. Also, the scheme gives a better result for security measures like RS Analysis, Chi-Square analysis, and NCC, which proves the robustness of the stego image against such attacks.

3.1 SS Using Graph Neighbourhood (SSGN)

The embedding capacity and perceptibility are the important criteria of a steganographic scheme. Taking these criteria into account, a steganographic scheme is designed using the concept of graph theory that uses a color image as a cover object. Images are partitioned into $(n \times n)$ sized non-overlapping pixel blocks. Every $(n \times n)$ pixel block is considered as a separate graph where pixels are the nodes of the graph. A Pixel Category Block (PCB) is formed from the pixels of the pixel block. A range table is considered, and the category number of a pixel is calculated according to its pixel value. The block of the degree of neighbourhood of the pixels is then formed. The degree of neighbourhood of the pixels are calculated according to the graph neighbourhood degree method [33]. A threshold value is then considered, and the pixels whose degree of neighbourhood is equal or more than the threshold value are selected. These selected pixels are considered for secret data embedding. Instead of embedding secret data in all these selected pixels, a particular pixel is selected using a weighted matrix to embed data. Selecting similar pixels for data embedding and introducing a weighted matrix improves the embedding capacity as well as enhances the visual quality of the stego image compared to some other existing state-of-the-art methods. Finally, the proposed scheme is tested against different steganographic attacks, and various analysis are performed to observe its imperceptibility and robustness.

3.1.1 Data Embedding Procedure

In this section, a novel graph-based steganographic technique using a weighted matrix has been proposed. The embedding process is depicted using a schematic diagram in Fig. 3.1. At first, a color image (C) of size $(m \times n)$ is taken as the cover image. A secret image (S) is taken and then it is converted into a secret bitstream (BS). The cover image is then partitioned into (4×4) pixel blocks (PB). Each pixel block (PB) is considered as a graph where the pixels are the nodes of the graph. A range table is maintained to categorize the pixels according to their values. A sample range table is shown in Fig. 3.1. A sample range table is shown in Fig. 3.1. The gap between the upper limit of a range and the lower limit of the next range is 2. It ensures that a pixel category does not change due to addition or subtraction of a pixel value by 1. According to the given range table, a pixel with value 30 is of category 1, and a

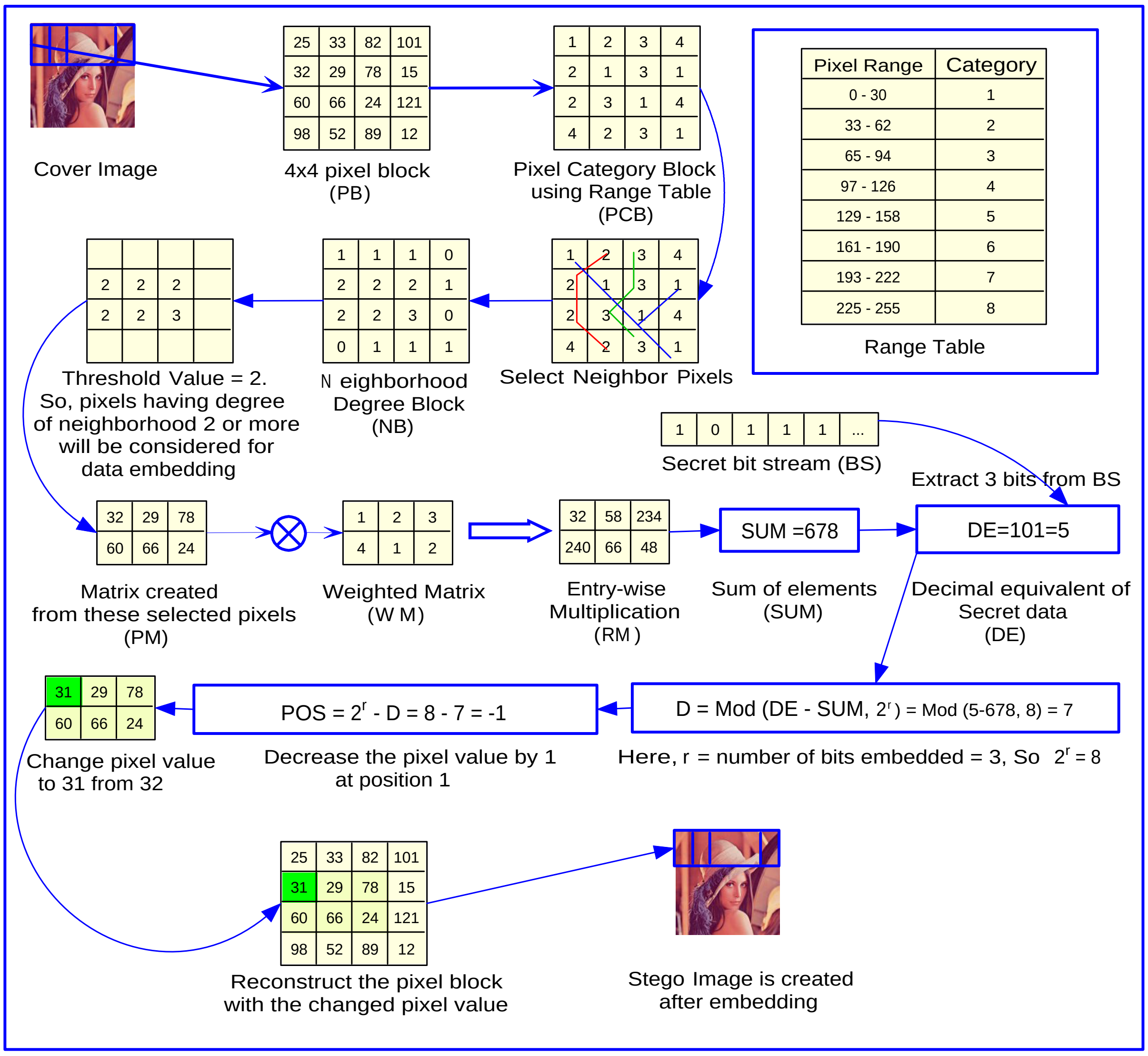

Cover Image
4x4 pixel block (PB)
Pixel Category Block using Range Table (PCB)
Pixel Range | Category
0 - 30 | 1
33 - 62 | 2
65 - 94 | 3
97 - 126 | 4
129 - 158 | 5
161 - 190 | 6
193 - 222 | 7
225 - 255 | 8
Range Table
Select Neighbor Pixels
Neighborhood Degree Block (NB)
Threshold Value = 2. So, pixels having degree of neighborhood 2 or more will be considered for data embedding
Secret bit stream (BS)
Extract 3 bits from BS
Matrix created from these selected pixels (PM)
Weighted Matrix (W M)
Entry-wise Multiplication (RM)
SUM =678
Sum of elements (SUM)
DE=101=5
Decimal equivalent of Secret data (DE)
D = Mod (DE - SUM, 2^r) = Mod (5-678, 8) = 7
Here, r = number of bits embedded = 3, So 2^r = 8
POS = 2^r - D = 8 - 7 = -1
Decrease the pixel value by 1 at position 1
Change pixel value to 31 from 32
Reconstruct the pixel block with the changed pixel value
Stego Image is created after embedding

Figure 3.1: *Schematic diagram of data embedding procedure in SSGN*

pixel with value 130 is of category 5 etc. A PCB is formed from the categories of the pixels of the PB. In a PCB, each pixel or node may have other pixels around it having the same category or group number. The number of pixels around any pixel having the same category number is its neighbourhood degree. A (4×4) neighbour degree block (NB) is formed. The value 2 in a NB means that it has 2 adjacent pixels of the same category, a value of 3 means that it has 3 adjacent pixels of the same category. Before embedding, a threshold value between 0 to 8 is chosen to select the pixels where the secret data will be embedded. For example, if the threshold value is 2, then all the pixels having neighbourhood degree with 2 and above are selected for data embedding. Once the pixels are selected from a pixel block, a $(m \times n)$ Pixel Matrix (PM) is formed. A Weighted Matrix (WM) is multiplied with the PM using entry-wise multiplication to form the Resultant Matrix RM. The sum of the elements (SUM) of the

Algorithm 3.1: SSGN: Embedding Algorithm

```
input : A Cover Image of size $m \times$n, Secret message
output: A Stego Image of size $m \times$n
Algorithm Embedding():
    for (pixelBlock ← 1 to $(m \times n)/4$ ) do
        Step-1: A matrix $SM_{x \times y}$ is formed using the selected pixels from the $4 \times 4$ $pixelBlock$ of cover image using the following function
        CreatePixelMatrix();
        Step-2: The secret data is embedded in the selected pixels using the following function
        EmbedSecretData();
    Step-3: The stego image is generated after embedding the secret bits
Function CreatePixelMatrix():
    Step-1: Get neighbourhood degree of all pixels in $4 \times 4$ block of the cover image
    Step-2: A threshold value is considered here
    Step-3: If the neighbourhood degree of a pixel is more than or equals to the threshold value, add the pixel in the $SM_{x \times y}$ matrix
Function EmbedSecretData():
    Step-1: An weighted matrix ($WM_{x \times y}$) having the same dimension of $SM_{x \times y}$ matrix is formed
    Step-2: An elementary multiplication of $SM_{x \times y}$ with $WM_{x \times y}$ is performed
    Step-3: The sum of elements ($SUM$) of the resultant matrix is calculated.
    for $i$ ← 1 to x do
        for $j$ ← 1 to y do
            SUM $+= SM[i][j] \times WM[i][j]$
    Step-4: The $r$ number of bits are collected from secret bit stream and converted to its decimal equivalent $DE$
    Step-5: The difference $D$ is calculated by $D = MOD(DE - SUM, 2^r)$
    Step-6: The embedding position ($POS$) is the absolute value of $2^r - D$
    Step-7: The secret data is embedded by changing the value of pixel of the pixel block $PB$ in the position $POS$
    Step-8: The pixel value of $PB$ at position $POS$ is decreased or decreased by 1 depending on the value of $POS$ is negative or positive respectively
    Step-9: If the $POS$ value is zero, no change is made to the pixels of the pixel block $PB$
```

matrix RM is calculated. Then, r bits are collected from secret bit stream BS and converted to its decimal equivalent (DE). The difference D is calculated as: $D = MOD(DE - SUM, 2^r)$. The embedding position is the absolute value of POS which is calculated as $POS = 2^r - D$. No change is made to the pixels of the pixel block PB if the POS value is zero. Otherwise, the pixel value at position POS is decreased by 1 if the POS value is negative or increased by 1 if the POS value is positive. In this way, n bit secret data are embedded into a pixel block PB by modifying only a single pixel of the pixel block. After embedding all the secret data, a $(m \times n)$ stego image (SI) is generated. The embedding procedure is presented in Algorithm 3.1. In this algorithm $CreatePixelMatrix()$ function creates a matrix of pixels selected from PB depending on their degree of neighbourhood is greater than or equals to the threshold value. The $EmbedSecretData()$ function embed bits in this selected pixel matrix.

3.1.2 Data Extraction Procedure

The extraction process is depicted in Fig. 3.2. At first, the stego image (SI) of size $(m \times n)$ is taken for extraction of secret data. The SI is then partitioned into (4×4) non-overlapped pixel blocks (PB). Another range table has been used to categorize the pixels according to their values. A sample range table for extraction is shown in Fig. 3.2. Here, there is no gap between

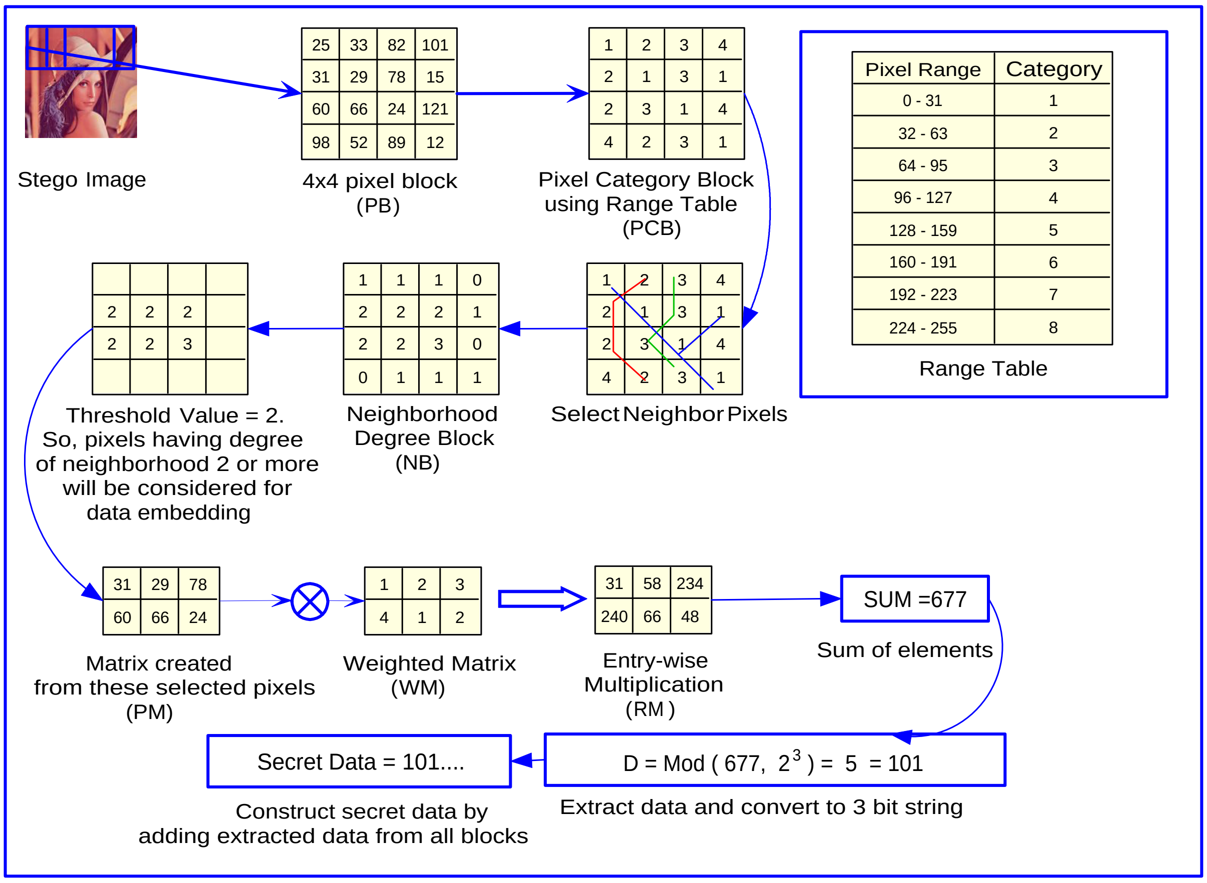


Figure 3.2: *Illustration of hidden data retrieval and cover image recovery in SSGN*

the upper limit of a range and the lower limit of the next range. A Pixel Category Block (PCB) is formed from the categories of the pixels of the PB. A (4×4) neighbour degree block (NB) is formed. After that, the same threshold value, used during embedding, is chosen to select the pixels from where the secret data will be extracted. Once the pixels are selected from a PB, a $(m \times n)$ Pixel Matrix (PM) is formed with the selected pixels. The WM is multiplied with the matrix PM using entry-wise multiplication to form the Resultant Matrix (RM). The sum of the elements (SUM) of the matrix RM is calculated. The modulo of SUM is then calculated as $D = MOD(SUM, 2^r)$. This D is actually the extracted data and is converted to r bit binary string. Secret bits from all other pixel blocks are extracted and then merged to form the whole secret data. Finally, the data is used to generate the secret image. The extraction procedure is presented in Algorithm 3.2. The $CreatePixelMatrix()$ function creates a matrix of pixels selected from PB depending on their degree of neighbourhood is greater than or equals to the threshold value. The $ExtractSecretData()$ function extracts secret bits from this selected pixel matrix.

Algorithm 3.2: SSGN: Extraction Algorithm

```
input  : A Stego Image of size $m \times$ n
output: Extracted Secret Message
Algorithm Extraction():
    for (pixelBlock ← 1 to $(m \times n)/4$ ) do
        Step-1: A matrix $SM_{x \times y}$ is formed using the selected pixels from the $4 \times 4$ $pixelBlock$ of cover image using the following function
        CreatePixelMatrix();
        Step-2: The secret data is extracted from the selected pixels matrix $SM$ using the following function
        ExtractSecretData();
    Step-3: The above steps are repeated for all pixel blocks of the color components to extract the secret bits from all pixel blocks
    Step-4: These secret bits are merged and the secret message is generated
Function CreatePixelMatrix():
    Step-1: Get neighbourhood degree of all pixels in $4 \times 4$ block of the cover image
    Step-2: The same threshold value, used during embedding, is considered here
    Step-3: If the neighbourhood degree of a pixel is more than or equals to the threshold value, add the pixel in the $SM_{x \times y}$ matrix
Function ExtractSecretData():
    Step-1: An weighted matrix ($WM_{x \times y}$) having the same dimension of $SM_{x \times y}$ matrix is formed
    Step-2: An elementary multiplication of $SM_{x \times y}$ with $WM_{x \times y}$ is performed
    Step-3: The sum of elements ($SUM$) of the resultant matrix is calculated.
    for $i$ ← 1 to x do
        for $j$ ← 1 to y do
            SUM $+= SM[i][j] \times WM[i][j]$
    Step-4: The modulo of $SUM$ is calculated as $D = MOD(SUM, 2^r)$
    Step-5: This $D$ is actually the extracted secret data. It is converted to $r$ bit binary string
```

3.1.3 Experimental Results and Comparisons

The scheme is tested using standard benchmark images shown in Fig. 1.1 and evaluated using various evaluation metrics. The experimental results and comparisons are given below:

3.1.3.1 Quality Measurement Analysis

The SSGN scheme is evaluated through some standard quality measurement analysis like PSNR, SSIM, and Q-Index using the equation (1.2), (1.4), and (1.10) respectively. In this SSGN scheme, the quality has been measured by PSNR. The PSNR obtained is 51 (dB) after hiding $7, 77, 965$ bits (approx.) in a cover image of size (512×512). Q-Index is used to measure the distortion of a stego image. The Q-Index of the proposed SSGN scheme is nearer to 0.9999. The similarity between two images is measured by SSIM whose values lie between -1 and +1. When the images are identical, SSIM value approaches to +1. In the proposed scheme, the SSIM is presented in Table 3.1. The average SSIM value is around 0.9960. The experimental results of PSNR (dB), SSIM, and Q-Index have been enlisted in Table 3.1 for different threshold values. It indicates good concealment of secret data in the stego image. When the threshold value is increased, that means the neighbourhood degree of the pixel has been increased, the data embedding capacity decreases and quality of a stego image increases. For example, when the threshold value is 2, the data hiding capacity of the Lena image is 742598 bits with PSNR

Table 3.1: *Results of evaluation metrics using standard images for varying threshold values from 1 to 8 in SSGN*

Threshold Value	Images	Capacity (in bit)	PSNR (in dB)	SSIM	NCC	BER	Q-Index
1	Lenna	777965	51.2190	0.9967	0.9998	0.0206	0.9998
	Airplane	780457	51.1755	0.9952	0.9999	0.0218	0.9997
	Boat	772988	51.2065	0.9929	0.9997	0.0207	0.9997
	Baboon	760836	51.2801	0.9962	0.9999	0.0209	0.9998
	Peppers	778013	51.1887	0.9969	0.9996	0.0210	0.9999
2	Lenna	742598	51.4090	0.9968	0.9999	0.0196	0.9999
	Airplane	754238	51.3315	0.9969	0.9998	0.0190	0.9998
	Boat	711640	51.5763	0.9968	0.9999	0.0192	0.9998
	Baboon	648246	51.9920	0.9968	0.9997	0.0193	0.9997
	Peppers	740201	51.4005	0.9969	0.9998	0.0196	0.9999
3	Lenna	678760	51.7992	0.9971	0.9999	0.0179	0.9998
	Airplane	706011	51.6101	0.9972	0.9999	0.0178	0.9999
	Boat	614733	52.2121	0.9971	0.9998	0.0178	0.9997
	Baboon	485322	53.2387	0.9973	0.9999	0.0175	0.9999
	Peppers	678381	51.7945	0.9972	0.9998	0.0178	0.9996
4	Lenna	547387	52.6548	0.9980	0.9999	0.0124	0.9999
	Airplane	584402	52.4284	0.9981	0.9997	0.0122	0.9998
	Boat	471350	53.3725	0.9980	0.9998	0.0124	0.9998
	Baboon	320622	55.0259	0.9983	0.9999	0.0123	0.9997
	Peppers	550590	52.6764	0.9981	0.9999	0.0120	0.9999
5	Lenna	411864	53.8798	0.9985	0.9999	0.0088	0.9999
	Airplane	461269	53.4563	0.9986	0.9998	0.0089	0.9996
	Boat	335999	54.8394	0.9984	0.9999	0.0088	0.9999
	Baboon	193552	57.2344	0.9985	0.9997	0.0087	0.9998
	Peppers	419480	53.8422	0.9986	0.9998	0.0088	0.9998
6	Lenna	248639	56.0739	0.9988	0.9997	0.0068	0.9997
	Airplane	289907	55.4699	0.9989	0.9998	0.0067	0.9999
	Boat	194866	57.2114	0.9989	0.9999	0.0068	0.9999
	Baboon	099335	60.1067	0.9987	0.9999	0.0069	0.9996
	Peppers	255369	55.9925	0.9988	0.9999	0.0068	0.9999
7	Lenna	120474	59.2170	0.9994	0.9998	0.0033	0.9999
	Airplane	141291	58.5883	0.9992	0.9999	0.0032	0.9999
	Boat	093319	60.3959	0.9993	0.9998	0.0033	0.9998
	Baboon	043501	63.7555	0.9994	0.9999	0.0034	0.9999
	Peppers	125657	59.0981	0.9991	0.9999	0.0033	0.9998
8	Lenna	094017	60.2858	0.9998	0.9999	0.0005	0.9999
	Airplane	125437	59.1031	0.9999	0.9999	0.0008	0.9999
	Boat	066855	61.8319	0.9997	0.9999	0.0005	0.9999
	Baboon	025520	66.0408	0.9999	0.9999	0.0007	0.9999
	Peppers	097995	60.1561	0.9999	0.9999	0.0006	0.9999

51.40 (dB). But, when the threshold value is 8, the hiding capacity of the Lena image is 94017 bits with PSNR 60 (dB). So, it is observed that when similar values of pixels in a PB are increased, the hiding capacity is decreased, but the quality is increased.

The proposed SSGN scheme has been compared with existing graph-based schemes of Ou et al. [92, 95], Li et al. [93], Peng et al. [94], Qu and Kim [96] and Sengul Dogan [33]. The com-

Table 3.2: *Comparison of SSGN with some state-of-the-art schemes in terms of PSNR (dB) when payload is 10000 bits*

Images	Ou et al.'s method [92]	Li et al.'s method [93]	Peng et al.'s method [94]	Ou et al.'s method [95]	Qu and Kim's method [96]	Sengul Dogan [33]	SSGN Scheme
Lena	59.6	60.3	60.5	60.6	60.3	65.4	69.9424
F16	63.7	62.0	62.9	63.3	63.7	65.2	70.1055
Baboon	55.2	53.5	53.6	54.5	54.2	65.4	70.0527
Barbara	59.4	59.8	60.1	60.2	59.8	65.3	70.0622
Boat	57.4	58.1	58.3	58.2	58.4	65.3	70.0253
Peppers	56.1	58.9	59.0	59.2	58.8	65.3	70.0006
House	64.8	61.8	64.4	63.7	64.6	65.2	70.1151
Sailboat	58.6	58.2	58.9	59.4	59.8	65.3	70.0253
Elaine	58.0	56.8	57.3	57.4	58.7	65.2	70.0123
Tiffany	58.0	60.1	60.6	60.3	60.6	65.3	70.0168
Average	59.2	58.9	59.6	59.7	59.9	65.3	70.0358

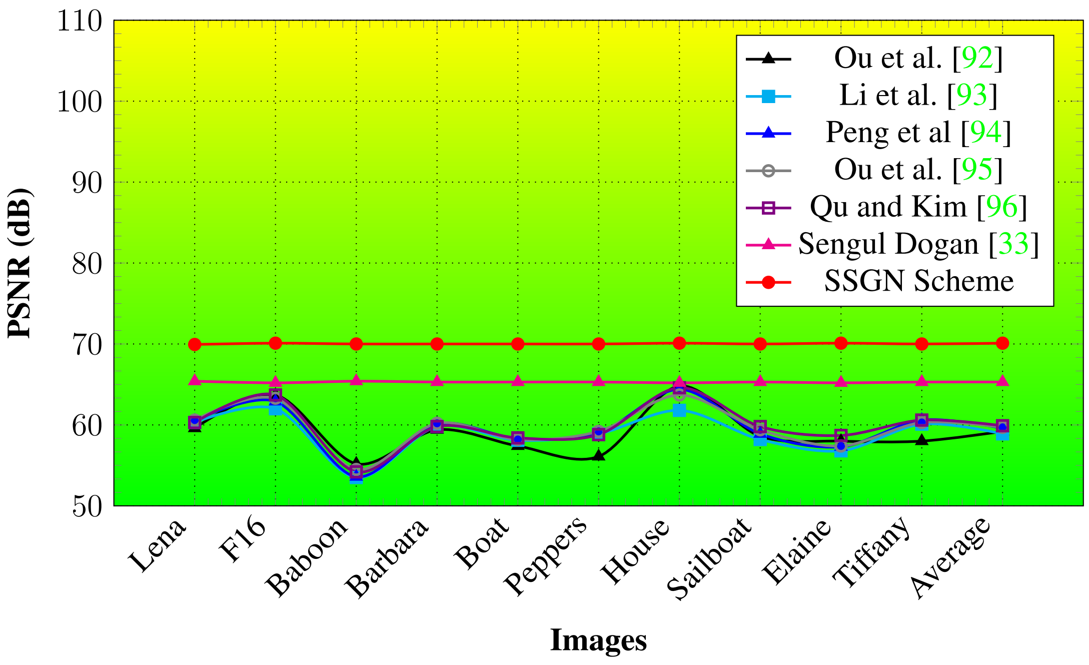


Figure 3.3: *Comparison graph of SSGN with existing schemes in terms of PSNR (dB)*

parison with respect to PSNR (dB) is given in Table 3.2 and Table 3.3 when embedding capacity is 10,000 and 20,000 bits respectively. It is observed that SSGN scheme achieves higher PSNR 70 dB and 67 dB when it carries 10,000 and 20,000 bits respectively. It is concluded that to achieve high embedding capacity with good visual quality, SSGN scheme performs better due to application of WM, which can hide more data bits by changing only one pixel in the selected pixel block.

The comparison of SSGN scheme with other existing schemes is graphically represented in Fig.

3.3. From Fig. 3.3 it is clear that the scheme outperforms other existing schemes after embedding 10,000 bits in cover image.

Fig. 3.4 represents the comparison of visual quality with different data embedding capacity.

Table 3.3: *Comparison of SSGN with some state-of-the-art schemes in terms of PSNR (dB) when payload is 20000 bits*

Images	Ou et al.'s method [92]	Li et al.'s method [93]	Peng et al.'s method [94]	Ou et al.'s method [95]	Qu and Kim's method [96]	Sengul Dogan [33]	SSGN Scheme
Lena	56.2	56.2	56.5	56.6	56.7	62.3	66.9472
F16	60.1	58.1	59.0	59.3	59.9	62.2	67.0146
Baboon	53.2	51.4	50.9	51.6	51.8	63.3	67.2735
Barbara	56.1	54.7	55.2	55.9	55.6	62.3	67.0103
Boat	53.3	53.3	53.9	53.7	54.2	62.3	67.0253
Peppers	52.8	54.7	54.7	54.9	55.0	62.2	67.1008
House	61.6	5.4	59.3	58.5	61.6	62.3	67.0485
Sailboat	53.6	53.4	53.6	54.3	54.7	62.3	67.0351
Elaine	52.9	52.4	52.6	52.7	53.7	62.4	67.1012
Tiffany	56.4	56.3	56.7	56.7	57.2	62.3	67.0338
Average	55.9	55.0	55.7	55.9	56.5	62.3	67.0352

From Fig. 3.4 it is easy to understand that PSNR varies inversely with the embedding capacity. After embedding approx 7,70,000 bits secret data, the scheme gives 51 dB PSNR (approx.).

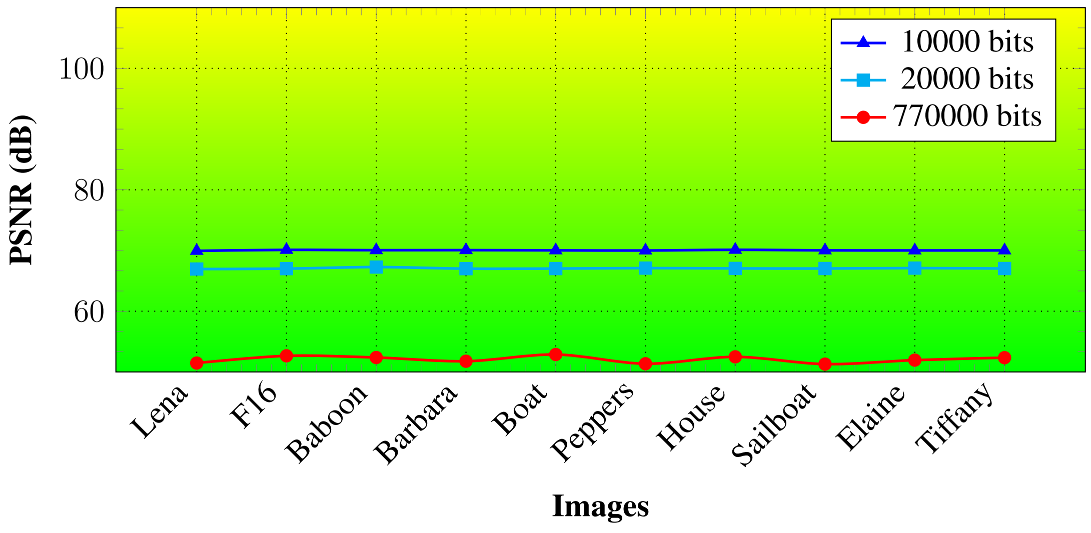


Figure 3.4: *Comparison graph of SSGN with varying data embedding capacity*

3.1.3.2 Robustness Analysis

NCC, BER, and RS-analysis are carried out to analyze the robustness of the proposed SSGN scheme using the equations (1.9), (1.11), and (1.5) respectively. The resultant value of NCC and

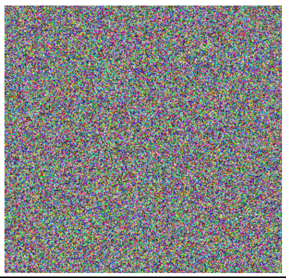
BER are shown in Table 3.1. The average NCC of all images lies between 0.9990 to 0.9999, which indicates high imperceptibility and robustness. The BER of the SSGN scheme is near 0.0206 when maximum secret data is embedded and 0.0006 when the average amount of secret data is embedded. The value of BER indicates that very less modification is made to the stego image while embedding secret data. Two standard image databases are used to perform RS analysis, which is mentioned in Table 3.4. The SSGN scheme has shown very good results against RS analysis. The average RS value of the SSGN scheme is near 0.0181 for SIPI image database [90] and 0.0196 for UCID image database [91]. The RS value near zero indicates that the SSGN scheme has less distortion in stego images after embedding secret data.

The security of the SSGN scheme depends on the weighted matrix, reference table, and threshold value. Some brute force attacks have been performed to show the security of the scheme. Fig. 3.5 shows three separate examples where a wrong weighted matrix, a wrong threshold value, and a wrong reference table is used to extract the secret image. The result proves that the attacker can only extract a noise-like image after using an invalid weighted matrix, threshold value, or reference table. In this context, it is clear that the SSGN steganographic scheme is highly robust and an authorized person, using valid weighted matrix, reference table and threshold value, can extract the secret message from the stego image without encountering any loss of data.

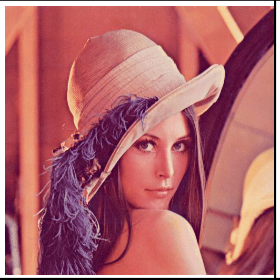
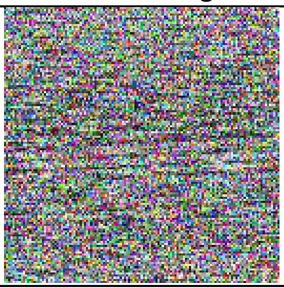
Table 3.4: *Results of RS analysis in SSGN*

Image Database	No. of images	Stego image				
		R_M	R_{-M}	S_M	S_{-M}	RS value
SIPI	05	23861	23572	20259	20467	0.0113
	20	22978	23120	17330	17173	0.0174
	50	33113	30676	06517	06778	0.0181
UCID	05	22354	22334	18137	18221	0.0026
	20	39959	43867	09524	08091	0.0079
	50	21411	21064	11942	12249	0.0196

(A) Attack with unknown weighted matrices

Original Image	Secret Image	Brute force attack	Extracted Secret Image
		1 2 3 4 5 6 8 7 5	
512×512	130×130	weighted matrix	PSNR= 7.9376(dB)

(B) Attack with unknown thresholds value

Original Image	Secret Image	Unknown Threshold Value	Extracted Secret Image
		Unknown Threshold = 2 Original Threshold =3	
512×512	130×130	Threshold Value	PSNR= 7.2154(dB)

(C) Attack with unknown Reference Table

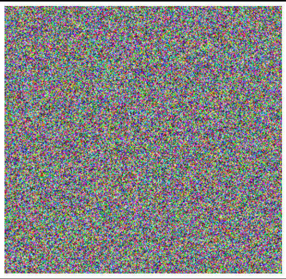
Original Image	Secret Image	Unknown Reference Table	Extracted Secret Image
		Range – Category 0-50 – 1 53-102 – 2 105-154 – 3 157-206 – 4 209-255 – 5	
512×512	130×130	Reference Table	PSNR= 7.5421(dB)

Figure 3.5: *Results of brute force attacks with unknown weighted matrix, threshold value and reference table in SSGN*

3.2 SS Using Pixel Value Difference (SSPVD)

A new color image based reversible steganographic scheme using Pixel Value Difference (PVD) has been proposed in this paper. Here, the cover image is generated by interpolating the pixels of the input color image. The cover image is partitioned into separate image blocks, and the PVD method is repeatedly applied to each block to embed a secret message. It has been seen that the proposed scheme (SSPVD) gives 1.48 bpp payload and around 45 PSNR (dB). The proposed scheme is fully reversible. So, both the original input image and the secret image can be successfully extracted from the stego image. The proposed method is tested using different steganographic attacks like RS analysis, Chi-square analysis, and NCC to show that the scheme is undetectable under these analyses and more robust than some other schemes of the same kind. This scheme provides good embedding capacity with high visual quality of the stego image. PSNR is calculated to show that the image quality of the proposed method is better in comparison to some existing steganographic schemes.

3.2.1 Data Embedding Procedure

The embedding process of the SSPVD scheme is depicted in Fig. 3.6. Here, the color cover image (C) is split into three separate RED, GREEN and BLUE color component blocks. Now, a (2×2) pixel block from the RED color component is considered. This (2×2) pixel block is then converted to (3×3) RED color pixel block using interpolation by equation 3.1. This is repeated for the rest (2×2) pixel blocks of the RED color component. This process is iterated for the GREEN and BLUE color components also to get the final interpolated color image. Now another color image is taken as a secret image (S). The secret message is formed by extracting the color bits from S and merging into a bit stream in a sequential order. The secret message is considered as $M = m_1|m_2|m_3|\dots m_n$, where m_i, (i=1 to n) is 4 bits secret message block. Now a (3×3) RED pixel block is taken from the interpolated image, where i,j = 0, 1, 2. Now the pixels of this (3×3) pixel block can be considered as $P_{i,j}$, $P_{i,j+1}$, $P_{i,j+2}$, $P_{i+1,j}$, $P_{i+1,j+1}$, $P_{i+1,j+2}$, $P_{i+2,j}$, $P_{i+2,j+1}$, $P_{i+2,j+2}$ as shown in Fig. 3.6. At first, a pixel pair $P_{i,j}$ and $P_{i,j+2}$ are considered. The absolute difference d_i is calculated as $d_i = |P_{i,j} - P_{i,j+2}|$. Then, 4 bits from the secret message bit stream is taken and converted to its decimal equivalent sd_i. The subrange of d_i from reference table is checked to get the lower bound l_i of it. Then, l_i is added to sd_i

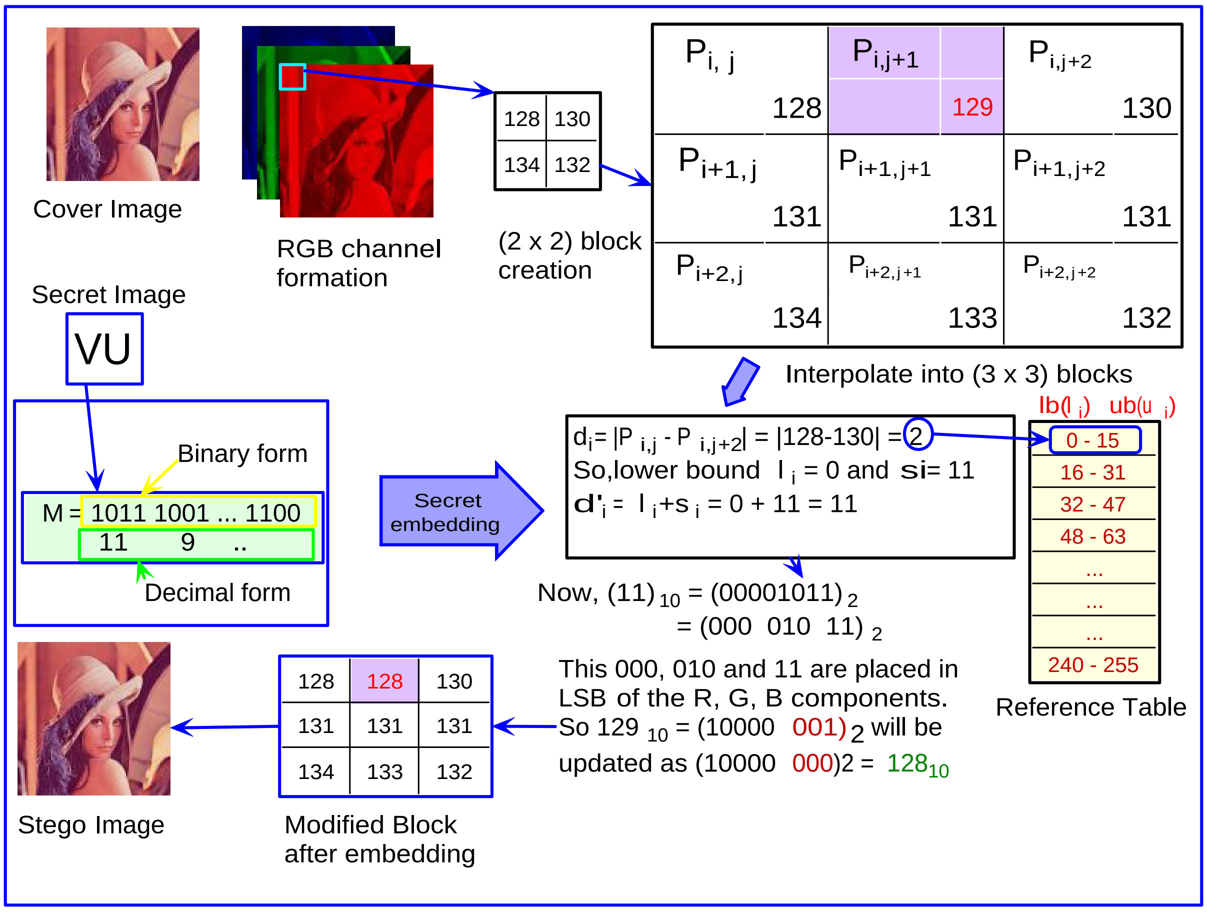


Figure 3.6: *Numerical illustration of embedding process in SSPVD*

to get d_i'. This d_i' is converted to 8 bits binary number. This 8 bit binary number is separated into three binary bit blocks of length 3,3 and 2. The first 3 bits are embedded into the 3 LSB of the RED color component of pixel $P_{i,j+1}$. The next 3 bits are embedded in the 3 LSB of the GREEN color component and the remaining 2 bits are embedded in the 2 LSB of the BLUE color component of the pixel $P_{i,j+1}$. Similarly, the pixel pairs $(P_{i,j}, P_{i+2,j})$, $(P_{i,j+2}, P_{i+2,j+2})$, $(P_{i+2,j}, P_{i+2,j+2})$ and $(P_{i,j}, P_{i+2,j+2})$ are considered to embed secret bits into the pixels $P_{i+1,j}$, $P_{i+1,j+2}$, $P_{i+2,j+1}$ and $P_{i+1,j+1}$ respectively. The same procedure is followed for the rest of the (3×3) pixel blocks of the interpolated image. Finally the stego image is created by all updated pixel blocks. The embedding procedure is given in Algorithm 3.3.

$$CI(i,j) = \begin{cases} C(p,q), \quad where\, p = 1,\dots,M; q = 1,\dots,N; \\ \frac{(C(i,j-1)+C(i,j+1))}{2}, \\ \quad where\ (i\, mod\, 2) = 0, (j\, mod\, 2) \neq 0; i = 1,\dots,(2\times M/3); j = 1,\dots,(2\times N/3) \\ \frac{(C(i-1,j-1)+C(i-1,j+1)+C(i+1,j-1)+C(i+1,j+1))}{4}, \\ \quad where\ (i\, mod\, 2) = 0, (j\, mod\, 2) = 0; i = 1,\dots,(2\times M/3); j = 1,\dots,(2\times N/3) \end{cases} \tag{3.1}$$

Algorithm 3.3: SSPVD: Embedding Algorithm

```
input  : Cover image $C_{(M \times N)}$, Secret image $S$
output: Stego image $SI$

Algorithm Embedding():
    Step-1: The color cover image of size $m \times n$ is converted to an 3d array of red, green and blue pixels
    Step-2: The interpolated image array is created using interpolation
    Step-3: The secret bits are collected to an array from the secret image
    Step-4: The secret bits are embedded into five positions of each $3 \times 3$ pixel block using the following method
    EmbedSecretBits();

Function EmbedSecretBits():
    // Perform the operation for each 3x3 image blocks
    // In each iteration embed secret bits in five interpolated pixel positions
    for (pixelBlock ← 1 to $(m \times n)/3$) do
        Step-1: The secret bits are embedded using the following method in each iteration
        EmbedBitsInPixel();

Function EmbedBitsInPixel():
    Step-1: Get two consecutive pixels
    Step-2: Get absolute value of their difference
    Step-3: Get the lower bound of the difference
    Step-4: Get decimal value of 4 bit secret data
    Step-5: Add this 4 bit decimal value to the lower bound
    Step-6: Convert the result value to a 8 bit binary string
    Step-7: Split this data into three parts
    Step-8: Convert first 3 bits, second 3 bits and last 2 bits separately to their decimal equivalent
    Step-9: Change LSB 3 of red pixel, LSB 3 of green and LSB 2 of blue pixel with these values respectively
```

3.2.2 Data Extraction Procedure

The data extraction procedure is presented in Fig. 3.7. At first the stego image (SI) is considered as an input image. It is partitioned into (3×3) pixel blocks. Now, the pixels of this (3×3)

pixel block can be considered as $P'_{(i,j)}$, $P'_{(i,j+1)}$, $P'_{(i,j+2)}$, $P'_{(i+1,j)}$, $P'_{(i+1,j+1)}$, $P'_{(i+1,j+2)}$, $P'_{(i+2,j)}$, $P'_{(i+2,j+1)}$, and $P'_{(i+2,j+2)}$ as shown in Fig. 3.7. First, the pixel $P'_{(i,j+1)}$ is considered. The pixel is separated into RGB color components. Then, the 2-LSB of BLUE component, 3-LSB of GREEN component and 3-LSB of RED component are collected and merged one after another to form a 8 bit binary number. The decimal equivalent of this binary number is stored into $d''_{(i,j+1)}$. The lower bound of the number $d''_{(i,j+1)}$ is taken from the reference table and deducted from $d''_{(i,j)}$ to get m'_1. This m'_1 is actually the secret bits extracted from the pixel $P'_{(i,j+1)}$. Then, in the similar way, the pixels $P'_{(i+1,j)}$, $P'_{(i+1,j+2)}$, $P'_{(i+2,j+1)}$ and $P'_{(i+1,j+1)}$ are considered to get m'_2, m'_3, m'_4 and m'_5 secret message bits. Then m'_1, m'_2, m'_3, m'_4 and m'_5 are merged one after another to get the total secret bits embedded into the first pixel block. The same procedure is followed for the rest of the pixel blocks to get the whole secret message. The secret image is reconstructed from these extracted secret bits. Now, the original (2×2) pixel block of the cover image is constructed by eliminating the $P'_{(i,j+1)}$, $P'_{(i+1,j)}$, $P'_{(i+1,j+2)}$, $P'_{(i+2,j+1)}$ and $P'_{(i+1,j+1)}$ pixels from the (3×3) pixel block of the input stego image. Then, the same procedure is followed for the rest of the (3×3) pixel blocks of the input stego image. Finally, the original cover image is reconstructed from all these recovered (2×2) pixel blocks. The extraction procedure is given in Algorithm 3.4.

Algorithm 3.4: SSPVD: Extraction Algorithm

```
input  : Stego image $SI$
output: Cover image $C_{(M \times N)}$, Secret image $S$
Algorithm Extraction():
    Step-1: Convert stego image to an 3 dimensional array, for red, green and blue pixels
    Step-2:Embed secret bits in every red, green and blue block of the interpolated image using the following method:
    ExtractSecretBits();
    Step-3: The secret image is constructed after extracting all the secret data
    Step-4: The cover image is formed after collecting all the unchanged pixels from stego image

Function ExtractSecretBits():
    // Perform the operation for each $3 \times 3$ image blocks
    // In each iteration extract secret bits from five pixel positions
    for (pixelBlock ← 1 to $(m \times n)/3$) do
        Step-1: Extract secret bits from five pixel positions by calling the following function in each iteration
        ExtractBitsFromPixel();

Function ExtractBitsFromPixel():
    Step-1: LSB 3 From RED pixel, LSB 3 from GREEN pixel and LSB 2 from BLUE pixel are extracted
    Step-2: These data are converted to 3 bit, 3 bit and 2 bit binary strings respectively and appended to form an 8 bit binary string
    Step-3: This 8 bit binary string is converted to decimal equivalent
    Step-4: The lower bound of this decimal data is calculated and subtracted from the decimal data
    Step-5: The result is converted to 4 bit string and append it to original extracted data
```

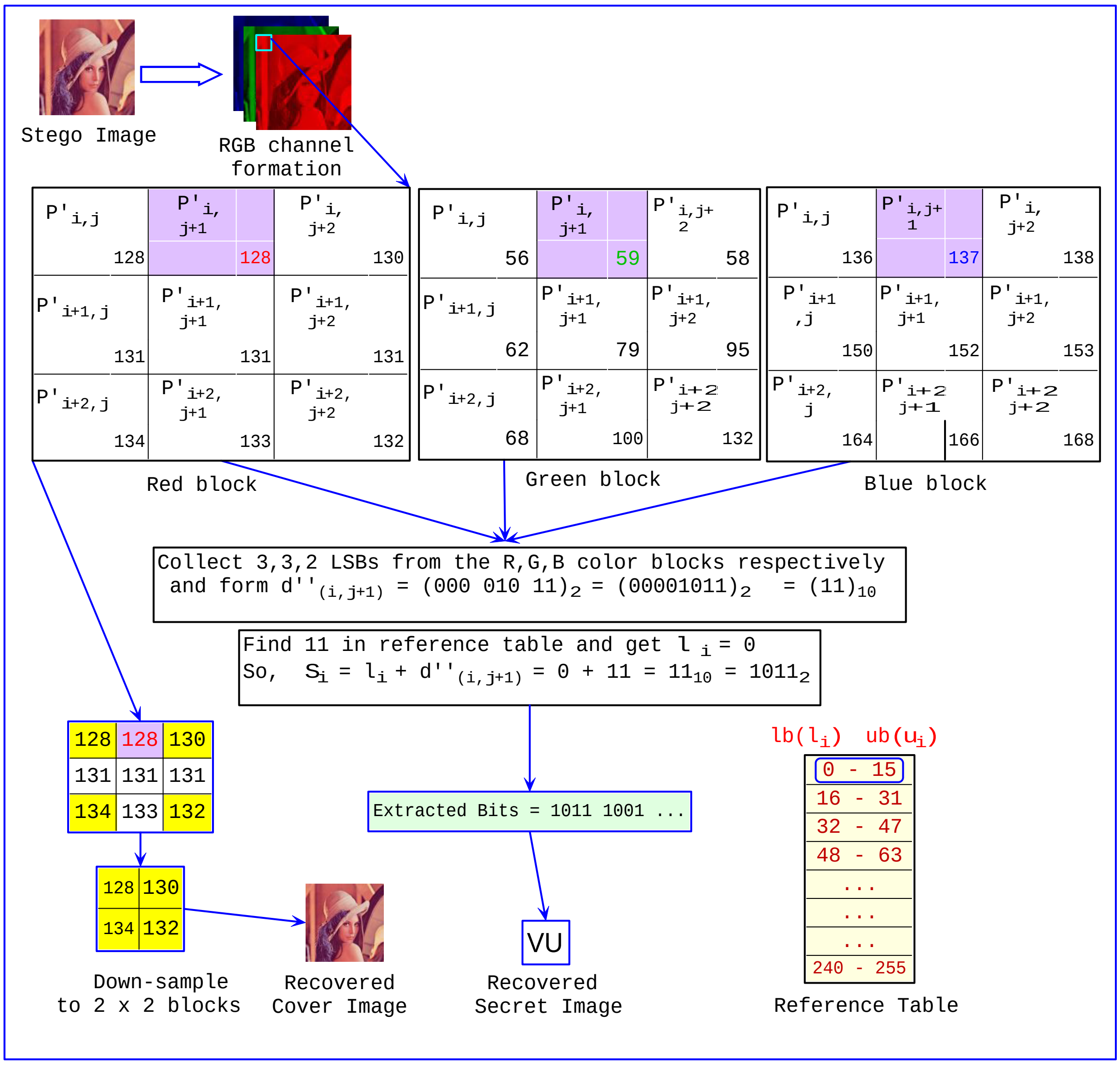


Figure 3.7: *Numerical illustration of extraction process in SSPVD*

3.2.3 Experimental Results and Comparisons

The proposed scheme has been tested using some standard benchmark images, as shown in Fig. 1.1 and evaluated using various evaluation metrics. The experimental results and comparisons are given below.

3.2.3.1 Quality Measurement Analysis

A logo image of size (330×330) has been used as a secret message. During the experiments, the same experimental procedure is applied to the proposed method and other methods ([3], [97], [5], [7], [98], [99]). The performance evaluation of the proposed scheme is evaluated using PSNR, SSIM, NCC, and Q-Index. The visual quality of the stego image is indicated by the PSNR defined by equation 1.2. The scheme is compared with some existing PVD based

Table 3.5: *Comparison table of proposed scheme with existing schemes in terms of PSNR (dB) and Capacity in SSPVD*

Cover Image	Wu and Tsai [3]		Zhang and Wang [97]		Wang et al. [5]		Joo et al. [7]		Shen et al. [98]		Jana et al. [99]		Proposed scheme (Color Image)		Proposed scheme (Grayscale Image)	
	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR	Capacity	PSNR
Lena	50,894	41.5	50,023	44.3	50,894	43.4	50,894	43.4	402485	42.46	9,16,656	37.17	2,621,440	45.25	9,83,040	44.37
Baboon	57,028	37.0	53,568	40.4	57,043	40.2	57,043	39.2	443472	38.88	9,16,656	37.17	2,621,440	45.32	9,83,040	44.82
Boat	52,320	39.6	50,926	42.4	52,490	41.1	52,490	41.0	408777	41.60	9,16,656	37.17	2,621,440	45.27	9,83,040	44.19
Elaine	50,981	42.1	49,750	44.5	50,893	44.9	50,893	43.5	398250	42.98	....	.....	2,621,440	44.76	9,83,040	44.09
Jet	51,221	40.6	50,366	43.5	51,221	43.4	52,662	41.5	......	.....	9,16,656	37.88	2,621,440	44.32	9,83,040	44.17
House	52,418	40.0	51,003	42.7	52,572	42.4	52,572	41.5	412354	41.16	9,16,656	37.18	2,621,440	44.02	9,83,040	43.87
Peppers	50,657	41.5	49,968	43.9	50,885	43.4	50,815	42.5	......	.....	9,16,656	37.17	2,621,440	43.79	9,83,040	43.18
Average	52,204	40.3	50,803	43.1	52,275	42.6	52,275	41.9	413067	41.42	9,16,656	37.28	2,621,440	44.67	9,83,040	44.09

state-of-the-art schemes in terms of PSNR and embedding capacity. The values are enlisted in Table 3.5. Though the scheme is designed for color images, it can also be applied for the grayscale image. In that case, the 3-LSB of the interpolated pixels of the grayscale image is changed for embedding data. In Table 3.5, PSNR, and capacity of both color image and grayscale image is given. From the table, it is observed that the average PSNR of the suggested scheme is around 44 dB, which is higher than Wu and Tsai's [3], Zhang and Wang's [97], Wang et al.'s Wang et al. [5], Joo et al.'s [7], Shen et al.'s [98] and Jana et al.'s [99] schemes. The maximum embedding capacity of the scheme is $2,621,440$ bits. So, the payload of the scheme is $= \frac{Total\ embedded\ bits}{(Row \times Col)} = \frac{2621440}{(768 \times 768 \times 3)} = 1.48\ bpp$, which is much higher than the existing state-of-the-art methods. The comparison graph of PSNR and embedding capacity is shown in Fig. 3.8. Figure 3.8 clarifies that, for other schemes, if the PSNR is high, then the capacity is low and if the capacity is high, then the PSNR is low. Only the proposed SSPVD scheme exhibits high PSNR even when the embedding capacity is reasonably high.

The SSIM, Q-Index, and NCC values of some test images from two image data sets are shown in Table 3.6. SSIM is a parameter for measuring the similarity between two images. Its value approaches to +1 when two images are identical. Here, the average SSIM value is calculated around 0.9960, which indicates that the stego image is almost identical to the cover image. The average NCC and Q-Index values lie between 0.9996 to 0.9999 and 0.9992 to 0.9999, respectively. These prove that there is minimum distortion in the stego image.

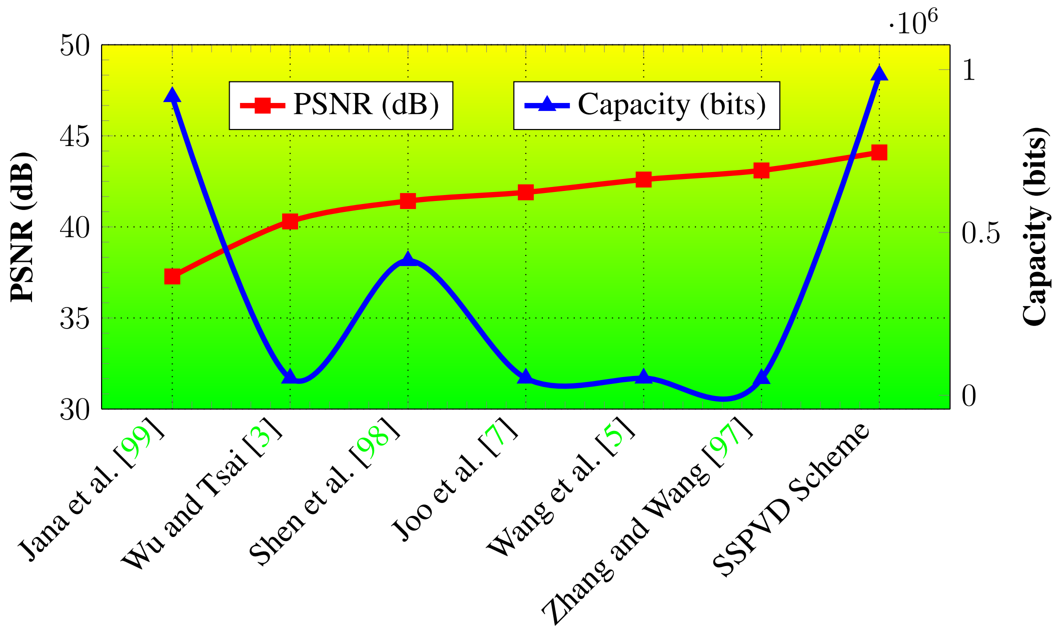


Figure 3.8: *Comparison graph with other existing schemes in terms of PSNR (dB) and embedding capacity*

Table 3.6: *Table for PSNR, SSIM, and NCC of different images in SSPVD*

Image Database	Cover Image	PSNR	SSIM	NCC	Q-Index
USC - SIPI Image Database (512 x 512)	Lenna	45.25	0.996	0.9999	0.99998
	Baboon	45.32	0.996	0.9998	0.99997
	Boat	45.27	0.996	0.9997	0.99996
	Elaine	44.76	0.996	0.9999	0.99992
	Jet	44.32	0.996	0.9998	0.99993
	House	44.02	0.996	0.9997	0.99994
	Peppers	43.79	0.996	0.9998	0.99995
	Soccer	44.96	0.996	0.9996	0.99996
	Zelda	43.25	0.996	0.9997	0.99997
	Barbara	44.32	0.996	0.9998	0.99998
UCID Image Database (512 x 512)	Ucid00008	48.3	0.996	0.9996	0.99996
	Ucid00009	47.6	0.996	0.9998	0.99997
	Ucid00011	45.3	0.996	0.9996	0.99999
	Ucid00013	48.6	0.996	0.9996	0.99998
	Ucid00015	48.3	0.996	0.9998	0.99994
	Ucid00058	47.6	0.996	0.9997	0.99995
	Ucid00061	45.3	0.996	0.9996	0.99996
	Ucid00062	48.6	0.996	0.9946	0.99993
	Ucid00128	48.6	0.996	0.9996	0.99998
	Ucid00166	44.7	0.996	0.9996	0.99994

3.2.3.2 Robustness Analysis

In this section, to verify the robustness and security of the proposed steganographic scheme, the stego image has been tested by some standard steganalysis technique like RS analysis, PVD histogram analysis, Standard Deviation (SD) and Correlation Coefficient (CC). Here the stego image is analysed through RS Analysis using the equation 1.5, and the values are shown in Table 3.7. From the results, it is clear that the stego images successfully passed the RS analysis for $R_m \cong R_{-m}$ and $S_m \cong S_{-m}$ which proves that no stego image will not be treated as a suspicious image. So, it is very difficult for the eavesdropper to detect the secret message hidden in the stego image, which proves that the proposed SSPVD scheme is secure against the RS detection attack.

PVD histogram may be a potential characteristic to expose the hidden message in the stego im-

Table 3.7: *Results of RS analysis in SSPVD*

Image	Data (Bits)	Stego image				
		R_M	R_{-M}	S_M	S_{-M}	RS value
Lena	2,621,440	59,214	52,959	39,696	45,951	0.1265
Baboon	2,621,440	36,406	33,213	25,374	28,378	0.1003
Boat	2,621,440	57,081	52,139	41,829	46,771	0.0999
Jet	2,621,440	59,807	53,035	39,103	45,875	0.1369
House	2,621,440	59,161	52,750	39,749	46,157	0.1296
Pepper	2,621,440	57,249	52,143	41,661	46,767	0.1032
Tiffany	2,621,440	56746	62,834	42164	36,076	0.1231
Car	2,621,440	57945	52,682	40965	46,228	0.1064
Barbara	2,621,440	56955	52302	41,955	46,608	0.0941
Zelda	2,621,440	58990	53727	39920	45183	0.1064

ages generated using the PVD based data hiding schemes. Fig. 3.9 shows the PVD histograms of the Lena image and its corresponding stego image with maximum embedding capacity. From this figure, it is observed that the PVD histograms can be well preserved after hiding secret data. Here, statistical analysis using standard deviation (SD) and correlation coefficient (CC) of the cover image (C) and stego image (SI) has been performed with the help of equation 1.8 and 1.7 respectively and the results are shown in Table 3.8. The SD of C and SI are 122.4311 and 122.4857 respectively, and their difference is 0.0546 for the Lena image. The CC between the image C and SI is 0.9995 for the Lena image, which implies that it is hard to locate the embedding position within the stego image (SI). Relative entropy, measured by the equation 1.6, is another security measurement tool. Number of secret bits decides the relative entropy of the cover image (C) and the stego image (SI). The relative entropy in the stego image increases with the increase of secret bits in the stego image. Relative entropy values of some standard benchmark images are listed in Table 3.9. From the table, it is clear that the difference between relative entropy values of C and SI images are very small, which implies that the SSPVD

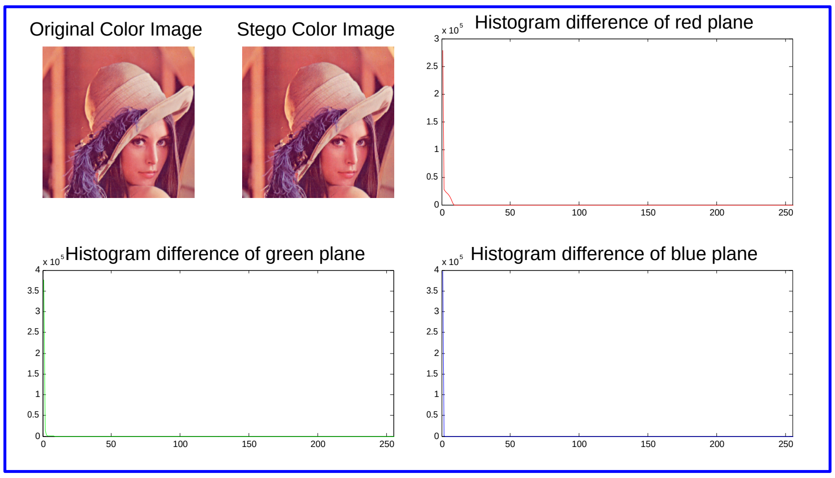


Figure 3.9: *Histogram difference of cover and stego image in SSPVD*

Table 3.8: *Standard deviation (SD) and correlation coefficient (CC) of cover image and stego image in SSPVD*

Cover Image	SD of cover image (C)	SD of Stego image (SI)	CC between C & SI
Lena	122.4311	122.4857	0.9995
Baboon	134.4063	134.3275	0.9996
Boat	196.9643	196.9843	0.9998
Jet	131.7815	131.8394	0.9996
House	177.9450	177.8902	0.9998
Pepper	123.2961	123.1225	0.9995
Tiffany	77.3370	76.6433	0.9986
Car	147.8972	148.0279	0.9997
Barbara	163.2217	163.2173	0.9998
Zelda	151.4704	151.3740	0.9997

scheme provides secure hidden communication.

Table 3.9: *Relative entropy between original image and stego image in SSPVD*

Image	Data(bits)	Entropy (C)	Entropy (SI)	Entropy Difference
Lena	2,621,440	7.4429	7.4622	0.0193
Baboon	2,621,440	6.6752	6.7044	0.0292
Boat	2,621,440	7.2988	7.3185	0.0197
Jet	2,621,440	6.7004	6.7252	0.0248
House	2,621,440	6.4568	6.4763	0.0195
Pepper	2,621,440	6.6251	6.6464	0.0213
Tiffany	2,621,440	6.5946	6.6142	0.0196
Car	2,621,440	6.4287	6.4488	0.0201
Barbara	2,621,440	6.5746	6.5875	0.0129
Zelda	2,621,440	6.6354	6.6565	0.0211

3.3 Analysis and Discussion

The two proposed steganographic schemes are compared and described below in Table 3.10. From this table, it is observed that the payload is better in PVD based scheme, whereas the visual quality of the image i.e. PSNR is better in the graph-based scheme. The visual quality is better in graph-based scheme because the data is embedded into a similar type of pixels in a pixel block. The payload is better in PVD based scheme because interpolated pixels are used here and PVD is applied repeatedly to embed data into interpolated pixels. The graph-based scheme is not reversible. But reversibility has been achieved in PVD based scheme with the help of interpolated pixels. The unknown weighted matrix and reference table play an important role in enhancing the security of the schemes. The stego images of these schemes are analysed through RS analysis. The relative entropy and the correlation coefficient (ρ) are calculated, and these are shown in Table 3.11.

Table 3.10: *Comparison of SSGN and SSPVD in terms of PSNR (dB) and payload (bpp)*

Schemes	Reversible/Irreversible	Capacity (bits)	PSNR (dB)	Payload (bpp)
SSGN	Irreversible	777965	51.22	0.989
SSPVD	Reversible	2621440	45.32	1.480

Table 3.11: *Comparison of steganalysis values of SSGN and SSPVD*

Schemes	RS Value	Relative Entropy	CC
SSGN	0.0382	0.0089	0.9762
SSPVD	0.0568	0.0072	0.9874

Key features of the schemes discussed in this chapter:

i) Pixel value difference (PVD) has been used in the interpolated pixels of a pixel block to improve the embedding capacity of the proposed scheme.

ii) Graph Neighbourhood based steganographic protocol has been developed to achieve high imperceptibility of the stego image.

iii) Any arbitrary length of the secret message can be communicated through graph-based Steganographic scheme.

iv) A weight matrix is used as a shared secret key where a list of random numbers is used to form the matrix.

v) A shared secret key is used in the PVD based scheme to maintain the sequence of secret data embedding.

The payload and the security of these schemes are moderate. Moreover, the graph-based steganographic scheme is not reversible. To achieve reversibility and to improve the payload without compromising the imperceptibility, two new dual image based reversible steganographic schemes have been proposed in the next chapter.