Chapter 2

Literature Review

2.1 Introduction

Steganography is an art of invisible communication. As per the cover medium is concerned, we have 4 variants of it. (1) video steganography, (2) audio steganography, (3) text steganography, and (iii) image steganography. Image steganography methodologies are of 2 types based on domain, (i) spatial, and (ii) transform. The steganography methodologies are characterized by three quality parameters like HC, distortion measure, and security.

The various spatial domain image steganography methodologies are, (i) LSB substitution, (ii) improved LSB based techniques, (iii) PVD techniques, (iv) LSB+PVD techniques, (v) modulus function (MF) based techniques, and (vi) Edge based techniques, etc. Steganalysis is the art of detecting steganography. The RS and PDH analyses are applied on LSB and PVD based methodologies. This Chapter describes the spatial domain methodologies, RS analysis, PDH analysis, and the three research problems.

2.2 A Study on Image Steganography Techniques

2.2.1 LSB Substitution Technique

Kurak and McHugh [20] started the concept of hiding in LSB positions in 1992. As per this principle as the LSBs are less dominant to decide the intensity value of the pixel, so replacing an LSB bit by a data bit will not result in much change in intensity value. One example of 1-LSB change is demonstrated in Fig 2.1. We can notice in this diagram that 1 secret bit is substituting a LSB of a pixel. Thus 3 bits are substituting LSBs of 3 different pixels. If we want to hide more number of bits, then we can use up to 4 LSBs. So, in this way we can accommodate huge number of bits in the image. But if you use the LSBs of all the pixels sequentially, the SI can be detected by RS test [21] and also chi-square test [22]. Therefore, we shall try to use the LSBs of different pixels in a non-continuous manner. Table 2.1 enumerates the change of pixel value in case of single LSB substitution. One can observe that the change 2n ↔2n+1 holds good. Table 2.2 says that 2n ↔2n -1 could not hold. The example in Table 2.1 shows that 62 ↔63 is possible. The example in Table 2.2 shows that 62 ↔61 is not possible. This idea is kept into a formal procedure and used as RS analysis.

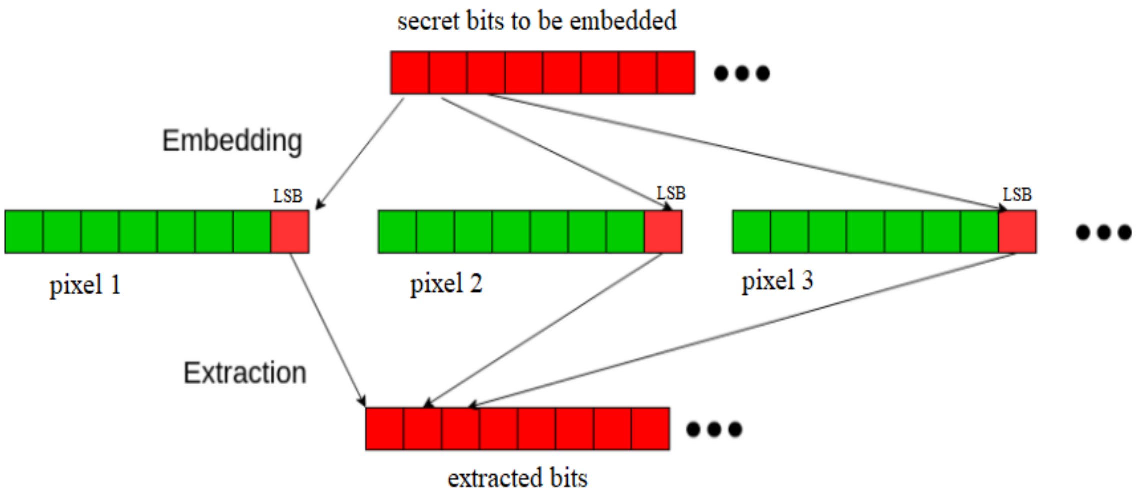


Fig. 2.1 An illustration of the 1LSB substitution

Table 2.1. Transformation 2n↔2n+1 (possible)

Pixel value (decimal)	Pixel value (binary)	Bit to be embedded	Stego-pixel value (binary)	Stego-pixel value (decimal)	Alteration
62=2×31=2n, here n=31	00111110	0	00111110	62=2×31	No change
		1	00111111	63=2×31+1	2n→2n+1
63=2×31+1=2n+1, here n=31	00111111	0	00111110	62=2×31	2n+1→2n
		1	00111111	63=2×31+1	No change

Table 2.2. Transformation 2n↔2n-1 (not possible)

Pixel value (decimal)	Pixel value (binary)	Bit to be embedded	Required value (decimal)	Required value (binary)	Alteration not possible
62=2×31=2n, here n=31	00111110	0	61=2×31-1	00111101	2n→2n-1 62→61
		1			
61=2×31-1=2n-1, here n=31	00111101	0	62=2×31	00111110	2n-1→2n 61→62
		1			

After it is known that RS test detects the LSB technique, researchers attempted to patch up in different ways to improve the security aspects. To achieve this Swain and Lenka [23] augmented cryptography with steganography. They also got larger HC. Here, they used encryption by a block cipher. However, they consumed only 2 bit planes out of 3 i.e., 6th, 7th and 8th bit planes.

The authors, Amirtharajan and Rayappan [24] used LSBs of the pixels in a random manner without going for continuous pixels, wherein the information is hidden in a block with least distortion. Undetectability is improved and PSNR is also good.

Word-hunt approach is proposed in [25] to reduce the expected number of changes in a pixel. This methodology accurately avoids the detections. Muhammad et al. also introduced an adaptive LSB approach with the help of a stego-key [26]. The secret data is enciphered by the cipher, and then buried in the OI. The results claim that this methodology maintains a good trade-off amid imperceptibility and HC. To bring up the HC, Wang et al. [27] adopted pixel adjustment without decreasing the image quality.

Authors in [28] brought an LSB replacement scheme by embedding information in k-LSBs. This technique produces quality SI, but is computationally complex. Chan et al. [29] suggested pixel adjustment after LSB alteration which generates good quality SI. Further, the merits of Chan et al.'s technique is the reduced time of execution. LSB steganography based on patterns is proposed to improve the security [113].

2.2.2 Improved LSB Based Techniques

The LSB array and LSB matching are improved LSB based techniques. Juneja and Sandhu [30] started LSB array-based embedding. The LSBs drawn from the pixels are grouped and known as LSB array. The secret data is converted to a bit-array. This bit-array is compared against the LSB arrays of numerous images, wherever maximum number of bits are matching, substituted there. But creating the LSB arrays of a larger number of images is not an easy task and it is very time consuming. Swain and Lenka [31] used multiple LSB arrays and selection of one array to hide the secret bits. Swain and Lenka [32] also introduced another variant i.e., binary word matching with LSB array. Here, every word is placed in a best fit place of the LSB array. It brings highest randomization and very hard to detect and steal the data. At the same time comparing every word against the LSB array is a time-consuming task.

Sharp [33] begun a new variant of LSB based approach known as LSB matching. In LSB matching, the LSB bits are not overlapped by data bits, rather the pixel value changes by ±1 arbitrarily. In fact, Mielikainen [34] implemented it in true sense. This way of concealing information provides better SI quality than the conventional LSB based schemes. Mielikainen's technique changes a pixel value in a random approach, so RS test cannot detect the SI. To get a clarity on this principle let us have a step-by-step description of Mielikainen's approach.

Step 1: Let us consider p_1, p_2, p_3,….,p_n as the pixels of the OI. Then, segregate 2 consecutive pixels as a block.

Step 2: Furthermore, let us say that the secret bits are s_1 and s_2 and (p_1, p_2) is a 2-pixel block.

Step 3: Camouflaging the bits is done in Eq. 2.1.

$$(p_1^*, p_2^*) = \begin{cases} (p_1, p_2), \text{ if } (\text{LSB}(p_1) = s_1) \text{ and } (f(p_1, p_2) = s_2) \\ (p_1, p_2 + 1), \text{ if } (\text{LSB}(p_1) = s_1) \text{ and } (f(p_1, p_2) \neq s_2) \\ (p_1 - 1, p_2), \text{ if } (\text{LSB}(p_1) \neq s_1) \text{ and } (f(p_1 - 1, p_2) = s_2) \\ (p_1 + 1, p_2), \text{ if } (\text{LSB}(p_1) \neq s_1) \text{ and } (f(p_1 - 1, p_2) \neq s_2) \end{cases} \quad (2.1)$$

Where f(p_1, p_2) = LSB of ($\lfloor p_1 / 2 \rfloor + p_2$).

Step 4: The process of extracting the camouflaged bits from (p_1^*, p_2^*) is very easy. LSB of p_1^* is retrieved as s_1, and s_2 can be gotten through Eq. 2.2.

$$s_2 = \text{LSB}\,(\lfloor p_1^* / 2 \rfloor + p_2^*) \qquad (2.2)$$

Improvements are done over the LSB matching technique by various researchers [35, 36, 37, 38] in terms of protection to security attacks.

In every pixel of a RGB image we have R, G, and B channels, each a byte. Each channel is 8 bits. We can consider, every channel as a gray pixel, where a LSB can be altered by a data bit. Hence a color pixel can hold 3 data bits. But RS test can crack it, so different variants [39, 40, 41, 42, 43, 44, 45, 46, 47] of RGB approach have been proposed. The main philosophy in these variants is that one of these 3 channels will be used as one indicator to specify HC for other 2 channels. Another philosophy is that the channels with higher values can accommodate higher changes.

2.2.3 PVD Techniques

As per LSB substitution, every pixel is treated equally, and in their LSBs data bits are held. Therefore, there is no discrimination between the edge regions and smooth regions as per this technique. In fact, researchers found that an edge region can hide more than a smooth region. Wu and Tsai's [49] PVD technique is based on this ideology. They took 2 consecutive pixels as a block and computed their difference. Replaced this difference by a new difference value and recomputed the pixel values. The new difference value holds the secret data bits.

Look at an example given in Fig.2.2. A block (P_i , P_{i+1})=(102,120). The difference d=18. It falls with range {16, 31}. In fact, the various ranges are {0,7}, {8, 15}, {16, 31}, {32, 63}, {64, 127}, and {128, 255}. For a range, suppose U signifies the upper bound (UB), and L signifies the lower bound (LB). For the d value 18, the L value is 16, and U value is 31. The block shall hide n bits. We shall compute n as $\lfloor \log_2(\text{U} - \text{L}) + 1 \rfloor$. Hence in this case n=4. The 4 data bits to be hidden are 1011_2. In decimal it is 11, so b=11. The new difference d' is computed as L+ b. Hence d' is 27. Again, compute m as $d_i' - d_i$. So, m is 9. Now by applying Eq. 2.3, the stego-block is (98, 125).

$$(P_i', P_{i+1}') = \begin{cases} \left(P_i' - \left\lfloor \frac{m}{2} \right\rfloor,\ P_{i+1}' + \left\lceil \frac{m}{2} \right\rceil\right), & \text{if d is even,} \\ (P_i' - \left\lceil \frac{m}{2} \right\rceil, P_{i+1}' + \left\lfloor \frac{m}{2} \right\rfloor), & \text{if d is odd} \end{cases} \qquad (2.3)$$

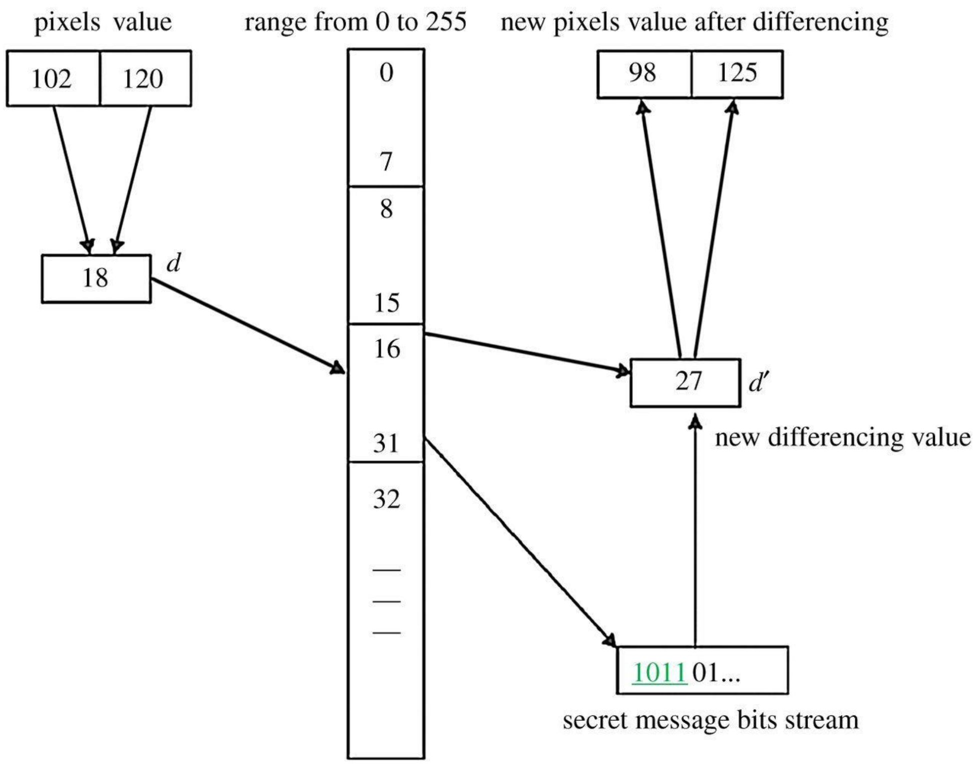


Fig.2.2 An example of PVD embedding

The above philosophy of Wu and Tsai is plied on 2×2 blocks to further raise the HC [50, 51]. But, the PDH test could detect the SIs of the above techniquese [52]. That is why the adaptive PVD (APVD) came into existence [9, 10].

There is no fixed range concept in APVD. The range is defined for every block afresh. Hence undetectability by PDH test is achieved. Huang [55] augmented mode-selection with 3-way PVD to bring multi-way PVD. It raised SI imperceptibility and improved the HC. Balasubramanian et al., [56] thought ahead in this direction and devised 8-PVD. By use of more edges this technique provided very large HC and also achieved a moderate value of PSNR. To get more security and get rid of sequential data embedding, Chen [57] used randomized hiding approach in 2×2 blocks and got undetectable SI. Hussain et al. [58] brought parity-bit based scheme to improve the undetectability. Tseng and Leng [59] used perfect square approach with 1×2 PVD to improve the HC and achieve undetectability. A good number of flavors are also brought with new thoughts to achieve improved performance in HC, PSNR and undetectability [60-63, 82,83].

Luo et al. [9] used 1×3 blocks for their APVD scheme. It is illustrated below. Let us consider a block in Fig.2.3(a). It has 3 pixels g_i, g_{i+1}, and g_{i+2}. From this block, we can compute 2 difference values plying Eq.2.4.

$$d_1 = g_{i+1} - g_i, \text{ and } d_2 = g_{i+2} - g_{i+1} \qquad (2.4)$$

If $|d_1| \leq T$ and $|d_2| \leq T$, then the block is left and no data is camouflaged in it, where T is a limit. Otherwise, we can embed in pixel g_{i+1} by computing its range. Ideally, we can consider the T value up to 10.

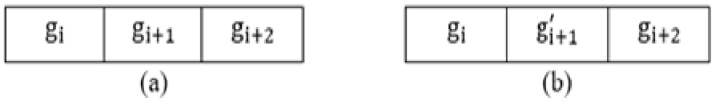


Fig.2.3 (a) 1×3 Original block and (b) 1×3 stego-block

Let us say that g'_{i+1} is the stego value of g_{i+1}. To hide in it its range is computed in the following 4 variants. The letter L is for LB and the letter U is for the UB.

Case 1: g_I is greater than g_{i+1}, and g_{i+2} is greater than g_{i+1}

If $|d_1| > T$ and $|d_2| > T$, then L=0, U=min(g_i-T-1, g_{i+2}-T-1)

If $|d_1| > T$ and $|d_2| \leq T$, then L=max(g_{i+2}-T, 0), U=min(g_i-T-1, g_{i+2}-1)

If $|d_1| \leq T$ and $|d_2| > T$, then L= max(g_i-T, 0), U=min(g_{i+2}-T-1, g_i-1)

Case 2: g_I is smaller than g_{i+1}, and g_{i+2} is smaller than g_{i+1}

If $|d_1| > T$ and $|d_2| > T$, then L= max(g_i+T+1, g_{i+2}+T+1), U=255

If $|d_1| > T$ and $|d_2| \leq T$, then L= max(g_i+T+1, g_{i+2}+1), U=min($g_{i+2} + T$, 255)

If $|d_1| \leq T$ and $|d_2| > T$, then L= max(g_{i+2}+T+1, g_i+1), U= min($g_i + T$, 255)

Case 3: $g_{i+1} \geq g_i$ and $g_{i+1} \geq g_{i+2}$

If $|d_1| > T$ and $|d_2| > T$, then L= g_{i+2}+T+1, U=g_i-T-1

If $|d_1| > T$ and $|d_2| \leq T$, then L= g_{i+2}, U=min($g_{i+2} + T$, g_i-T-1)

If $|d_1| \leq T$ and $|d_2| > T$, then L= max(g_{i+2}+T+1, g_i-T), U= g_i

Case 4: $g_{i+1} \leq g_i$ and $g_{i+1} \leq g_{i+2}$

If $|d_1| > T$ and $|d_2| > T$, then compute U=g_{i+2}-T-1, L= g_i+T+1

If $|d_1| > T$ and $|d_2| \leq T$, then compute U=g_{i+2}, L= max(g_{i+2}-T, g_i+T+1)

If $|d_1| \leq T$ and $|d_2| > T$, then compute U=min(g_{i+2}-T-1, g_i+T), L= g_i, After getting the values of U and L, the capacity, n is decided by Eq.2.5. here $k \leq 4$.

$$n=\min (k, \lfloor \log_2 |U - L + 1| \rfloor) \qquad (2.5)$$

Take n data bits and write it as decimal integer b. Ply Eq.2.6 to generate g'_{i+1}.

$$g'_{i+1} = \underset{e}{\operatorname{argmin}} \{|e - g_{i+1}| \mid |e - g_i| \equiv b \pmod{2^n},\ e \in [L, U]\} \qquad (2.6)$$

Fig.2.3(b) is your stego-block. Now we shall discuss about how to retrieve the data from this stego-block. To begin it, compute 2 difference values plying Eq.2.7.

$$d'_2 = g_{i+2} - g'_{i+1}, \text{ and } d'_1 = g'_{i+1} - g_i \qquad (2.7)$$

In case of $|d'_2| \leq T$ and $|d'_1| \leq T$, the block can be ignored. In other case we shall compute the UB and LB for $g'_{i+1}$by following the same 4 cases as we did in embedding procedure.

Thereafter, we compute n through Eq.2.5 and then b through Eq.2.8.

$$b \equiv |g'_i - g_u| \pmod{2^n} \qquad (2.8)$$

Now we convert b to n bits. Retrieval is done.

In Luo et al.'s scheme, the use of conditions $|d_1| \leq T$ and $|d_2| \leq T$ gives rise to "unused block problem (UBP)" and reduces the HC. That is why, to get more HC, Swain [10] used 2×2 and 3×3 blocks for APVD. Authors of [53] brought an optimized version of APVD steganography. It produces better PSNR with higher HC. To avoid the PDH test, Pradhan et al. [54] combined exploiting modification directions (EMD) with LSB and PVD schemes and achieved higher HC and PSNR as compared to Shen and Huang. To check the integrity of retrieved bits at receiver, an appropriate logic is provided in QVD and QVC based steganography [112].

2.2.4 LSB+PVD Techniques

LSB substitution results in higher HC. PVD results in better imperceptibility. By integrating both these thoughts in a single approach we can improve upon both the aims, HC and imperceptibility. Wu et al., [64] did this way. They did LSB in smooth blocks and PVD in non-smooth blocks. It is described below.

Step 1: The OI is broken up into 1×2 disjoint blocks. Suppose p_1 and p_2 denote the 2 pixels of one block.

Step 2: From p_1 and p_2, computer their difference v. If v ≤15, mark the block as smooth. If not, mark it as non-smooth (edge).

Step 3: In a non-smooth block, ply PVD idea of Wu and Tsai.

Step 4: In a smooth block, ply LSB substitution up to 3 bits for p_1 and p_2. Let p'_1 and p'_2 be the two new pixel values after 3LSB substitution. Then find v', difference amid p'_1 and p'_2. If $v' \leq 15$, then assign $p^*_1 = p'_1$ and $p^*_2 = p'_2$, otherwise apply Eq.2.9. Hence, p^*_1, and p^*_2 are obtained as stego-pixels.

$$(p^*_1, p^*_2) = \begin{cases} (p'_1 - 8, p'_2 + 8), \text{if } p'_1 \geq p'_2 \\ (p'_1 + 8, p'_2 - 8), \text{if } p'_1 < p'_2 \end{cases} \qquad (2.9)$$

Now we shall discuss about the data retrieval. To retrieve data from p^*_1 and p^*_2 compute, $v^* = |p^*_1 - p^*_2|$. In case of $v^* < 15$, extract 3 LSBs of both the pixels. Otherwise ply Wu & Tsai's extraction process.

In article [65] it is discussed that the technique in [64] is mostly satisfying v≤15, so doing mostly LSB replacement. So, it may be caught by RS analysis. Swain [66] considered a 3×3 blocks for plying LSB and PVD together. It performs well w.r.t HC and imperceptibility. Darabkh et al. [67] plied PVD with modified LSB substitution technique to raise the HC.

A LSB+PVD approach was also brought by Khodaei and Faez [68]. Their ideology is as follows. The OI is broken up into 1×3 blocks, as depicted at Fig.2.4(a). OI is considered as a 2D grid of pixels. In centre pixel g_x, k bit LSB change is done. Here, k may be 3, 4, or 5. Suppose g'_x is the stego-value (SV) of g_x. Again, let L is the decimal integer values for k LSBs of g_x, and S is the decimal integer values for k LSBs of g'_x. Denote d=L-S and compute g'_x through Eq.2.10.

$$g'_x = \begin{cases} g'_x + 2^k, \text{ if } d > 2^{k-1} \text{ and } 0 \leq g'_x + 2^k \leq 255 \\ g'_x - 2^k, \text{ if } d < -2^{k-1} \text{ and } 0 \leq g'_x - 2^k \leq 255 \\ g'_x, \text{ otherwise} \end{cases} \qquad (2.10)$$

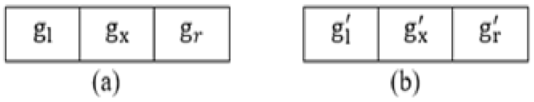


Fig. 2.4 Original and stego-blocks of size 1× 3

Now we shall calculate $d_1 = |g'_x - g_l|$ and $d_2 = |g'_x - g_r|$. We use a pair of range tables. For type 1, we use Table 2.3 and for type 2, we use Table 2.4.

Table 2.3 Khodaei and Faez’s Range table 1 (type 1)

Range	[0,7]	[8,15]	[16,31]	[32,63]	[64, 255]
capacity	3	3	3	4	4

Table 2.4 Khodaei and Faez’s Range table 2 (type 2)

Range	[0,7]	[8,15]	[16,31]	[32,63]	[64, 255]
capacity	3	3	4	5	6

Let l_1 is projected as LB and t_1 is projected as capacity of the range where d_1 fits in. Similarly, l_2 is projected as LB and t_2 is projected as capacity of the range of d_2. Represent the next t_1 data bits as s_1 and next t_2 data bits as s_2. Compute d'_1 and d'_2 plying Eq.2.11.

$$d'_1 = l_1 + s_1 \text{ , } d'_2 = l_2 + s_2 \qquad (2.11)$$

Plying Eq.2.12, g''_l, g'''_l, g''_r, and g'''_r are computed.

$$g''_l = g'_x - d'_1 \text{ , } g''_r = g'_x - d'_2 \text{ , } g'''_l = g'_x + d'_1 \text{ , } g'''_r = g'_x + d'_2 \qquad (2.12)$$

Plying Eqs. 2.13 and 2.14 g'_l and g'_r are computed. These are the final SVs for g_l and g_r respectively.

$$g'_l = \begin{cases} g''_l \text{ , if } 0 \le g''_l \le 255 \text{ and } |g_l - g''_l| < |g_l - g'''_l| \\ g'''_l \text{ , otherwise} \end{cases} \qquad (2.13)$$

$$g'_r = \begin{cases} g''_r \text{ , if } 0 \le g''_r \le 255 \text{ and } |g_r - g''_r| < |g_r - g'''_r| \\ g'''_r \text{ , otherwise} \end{cases} \qquad (2.14)$$

Hence Fig.2.4(b), represents the stego-block for Fig.2.4(a).

Now let us look at the data retrieval process. First, we divide SI into blocks as we did it earlier. Fig.2.4(b) is one block of SI. First, we extract LSBs from g'_x. After this use Eq.2.15 to compute d'_1 and d'_2.

$$d'_2 = |\, g'_r - g'_x \,|, \text{ and } d'_1 = |\, g'_l - g'_x \,| \qquad (2.15)$$

For d'_1, let the LB is l_1, and capacity is t_1. For d'_2, let the LB is l_2, and capacity is t_2. Tables 2.3 and 2.4 can be used as 2 range tables here also. Through Eq.2.16 calculate s_1 and s_2.

$$s_1 = d'_1 - l_1 \,,\; s_2 = d'_2 - l_2 \qquad (2.16)$$

Finally, s_1 and s_2 are converted to t_1 and t_2 binary bits accordingly. Extraction is done.

Gulve and Joshi [69] plied cryptography to add one more security layer in PVD+LSB approach. Hussain et al. [70] used prediction error based LSB+PVD approach and to make a trade-off amid HC and imperceptibility. Shukla et al. [71] augmented AES cryptography and compression logic with Khodaei and Faez's [68] to produces larger HC. Khodaei et al. [72] proposed an adaptive LSB+PVD mechanism. They made 2 adjoint pixels as a block. If any pixel value is more than 191, then 3LSB is changed for every pixel in that block. If not, compute the difference and trace it in a range table. From the range table, HC is known. Based on it pixel values are recomputed. This approach is not bad, it shows some superiority figures over other techniques.

Jung [73] used LSB+PVD methodology plying 2-bit planes to get higher HC. Let us consider a block with 2-adjacent pixels (P_1, P_2). As per this technique LSB camouflaging will be performed in k LSBs just by altering the LSBs and in rest (8-k) bits camouflaging will be performed plying PVD logic. Data camouflaging logic is given below. Compute remainders R_1=P_1 mod 2^k and R_2=P_2 mod 2^k. R_1 projects the integer value for k LSBs of P_1. R_2 projects the integer value for k LSBs of P_2. Similarly, compute Q_1=P_1 div 2^k and Q_2=P_2 div 2^k. For examples, 13 mod 3 = 1, and 13 div 3 = 4. This figures the decimal integer for remaining (8-k) bits of P_1, and P_2 accordingly.

Consider d=$|Q_1 - Q_2|$. Let L projects the LB and n projects capacity range of d in Table 2.5. Suppose the n data bits are denoted as S.

Table 2.5 Jung’s range table

Range	{0,7}	{8,15}	{16,31}	{32, 63}
capacity	3	3	4	5

Evaluate $d' = L+S$. Again, evaluate $m=|d' - d|$. Now ply Eq.2.17 to get (Q_1', Q_2').

$$(Q_1', Q_2') = \begin{cases} (Q_1 - \lceil m/2 \rceil\ ,\ Q_2\ +\ \lfloor m/2 \rfloor\ ), & \text{if d is odd} \\ (Q_1 - \lfloor m/2 \rfloor,\ Q_2\ + \lceil m/2 \rceil\ ), & \text{if d is even} \end{cases} \qquad (2.17)$$

Say, R_1' is integer value for next k secret bits. Then calculate $P_1' = Q_1' \times 2^k + R_1'$. Again, say R_2' is integer value for next k secret bits. Then compute $P_2' = Q_2' \times 2^k + R_2'$. Thus, P_1' and P_2' denote the SVs of P_1 and P_2 accordingly.

Now let us see how to retrieve the data from (P_1', P_2'). Calculate quotients and remainders through Eqs.2.18 and 2.19 accordingly.

$$Q_1' = P_1' \text{ div} 2^k \text{ and } Q_2' = P_2' \text{ div} 2^k \qquad (2.18)$$

$$R_1' = P_1' \bmod 2^k \text{ and } R_2' = P_2' \bmod 2^k \qquad (2.19)$$

Evaluate $d' = |Q_1' - Q_2'|$. Suppose L projects the LB and n projects capacity of a range of d' in Table 2.6. Evaluate $b = d' - L$. Write b in n bits. Project R_1' in k bits. Project R_2' in another k bits. Concatenate all these obtained bits to make a bit sequence. Extraction is done.

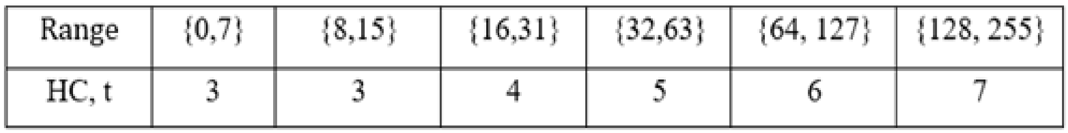
Pradhan et al. [83] integrated QVD with side match (SM) to obtain higher HC and PSNR. Liao et al. [74] brought a superior LSB+PVD approach to have good SI quality and HC. They decided the HC of a 4-pixel block by their average difference value. Kalita et al. [75] suggested XOR encoding with adaptive LSB and PVD to improve the performance. Swain [76, 77] used Khodaei and Faez’s [68] approach in 2×2 and 2×3 blocks and got higher PSNR. Various flavors of LSB+PVD approaches are available in recent literature [78-82].

2.2.5 Modulus Function (MF) Based Techniques

Wang et al. [84] initiated MF based PVD, wherein instead of the difference the remainder after the mod operator is replaced by a new value. Remainders are computed for 2 continuous pixels and they are changed to hold in the data bits. After this the distortion in SI is brought down using the optimization steps. Experimentally, it is worth to note that RS test could not detect the SI. Further, the FOBP is also avoided successfully. Wang et al., used 1×2 non-overlapped blocks to apply PVD. It is described here. Suppose, P_1 and P_2 are the 2 pixels of a block. The data hiding procedure has been narrated below.

Compute d=$|P_1 - P_2|$, it lies a range of Table 2.6, its HC is t.

Table 2.6 Wang et al.'s range table



Range	{0,7}	{8,15}	{16,31}	{32,63}	{64, 127}	{128, 255}
HC, t	3	3	4	5	6	7

Pick up t secret bits. Find its decimal equivalent as t'. Compute $F = (P_1 + P_2) \bmod 2^t$, $m = |F - t'|$ and $m_1 = (2^t - m)$. This t' is concealed inside P_1 and P_2 in any one of the below 8 cases to obtain stego-pixels P_1' and P_2' respectively.

Case 1: if $(F > t'$ and $m \leq 2^{t-1}$ and $P_1 \geq P_2)$, then $P_1' = P_1 - \left\lceil \frac{m}{2} \right\rceil$ and $P_2' = P_2 - \left\lfloor \frac{m}{2} \right\rfloor$

Case 2: if $(F > t'$ and $m \leq 2^{t-1}$ and $P_1 < P_2)$, then $P_1' = P_1 - \left\lfloor \frac{m}{2} \right\rfloor$ and $P_2' = P_2 - \left\lceil \frac{m}{2} \right\rceil$

Case 3: if $(F > t'$ and $m > 2^{t-1}$ and $P_1 \geq P_2)$, then $P_1' = P_1 + \left\lfloor \frac{m_1}{2} \right\rfloor$ and $P_2' = P_2 + \left\lceil \frac{m_1}{2} \right\rceil$

Case 4: if $(F > t'$ and $m > 2^{t-1}$ and $P_1 < P_2)$, then $P_1' = P_1 + \left\lceil \frac{m_1}{2} \right\rceil$ and $P_2' = P_2 + \left\lfloor \frac{m_1}{2} \right\rfloor$

Case 5: if $(F \leq t'$ and $m \leq 2^{t-1}$ and $P_1 \geq P_2)$, then $P_1' = P_1 + \left\lfloor \frac{m}{2} \right\rfloor$ and $P_2' = P_2 + \left\lceil \frac{m}{2} \right\rceil$

Case 6: if $(F \leq t'$ and $m \leq 2^{t-1}$ and $P_1 < P_2)$, then $P_1' = P_1 + \left\lceil \frac{m}{2} \right\rceil$ and $P_2' = P_2 + \left\lfloor \frac{m}{2} \right\rfloor$

Case 7: if $(F \leq t'$ and $m > 2^{t-1}$ and $P_1 \geq P_2)$, then $P_1' = P_1 - \left\lceil \frac{m_1}{2} \right\rceil$ and $P_2' = P_2 - \left\lfloor \frac{m_1}{2} \right\rfloor$

Case 8: if $(F \leq t'$ and $m > 2^{t-1}$ and $P_1 < P_2)$, then $P_1' = P_1 - \left\lfloor \frac{m_1}{2} \right\rfloor$ and $P_2' = P_2 - \left\lceil \frac{m_1}{2} \right\rceil$

If P_1' or P_2' lies in range {0, 255}, then it is fine. Otherwise, readjustment is done utilizing any one of the below 8 cases. This is performed to bring P_1' or P_2' value into the range {0, 255}.

Case 1: if $(P_1' < 0$ & $P_2' \geq 0,$ & $|P_1' - P_2'| > 128)$, then $P_1' = 0$ & $P_2' = P_1' + P_2'$

Case 2: if $(P_1' \geq 0$ & $P_2' < 0$ and $|P_1' - P_2'| > 128)$, then $P_1' = P_1' + P_2'$ & $P_2' = 0$

Case 3: if $(P_1' > 255$ and $P_2' \geq 0$ and $|P_1' - P_2'| > 128)$, then $P_1' = 255$ and $P_2' = P_1' + P_2' - 255$

Case 4: if $(P_1' \geq 0 \text{ and } P_2' > 255 \text{ and } |P_1' - P_2'| > 128)$, then $P_1' = P_1' + P_2' - 255$ and $P_2' = 255$

Case 5: if $(P_1' < 0 \text{ and } |P_1' - P_2'| \leq 128)$, then $P_1' = P_1' + 2^{t-1}$ and $P_2' = P_2'$

Case 6: if $(P_2' < 0 \text{ and } |P_1' - P_2'| \leq 128)$, then $P_1' = P_1'$ and $P_2' = P_2' + 2^{t-1}$

Case 7: if $(P_1' > 255 \text{ and } |P_1' - P_2'| \leq 128)$, then $P_1' = P_1' - 2^{t-1}$ and $P_2' = P_2'$

Case 8: if $(P_2' > 255 \text{ and } |P_1' - P_2'| \leq 128)$, then $P_1' = P_1'$, $P_2' = P_2' - 2^{t-1}$

The data retrieval is performed from 1×2 disjoint pixel blocks by the below procedure. In a 2-pixel block (P_1' , P_2'), d=| $P_1' - P_2'$| is computed. By referring to Table 2.6, the range and HC of the range t is identified. The embedded decimal digit, $t' = (P_1' + P_2') \bmod 2^t$, it is transformed to t bits. It is our retrieved data.

There is some irregularity in the PDH of Wang et al.'s scheme, so Joo et al. [85] addressed it plying turnover strategy. Li and He [86] advanced MFPVD by augmenting it with particle swarm optimization (PSO). Xu et al. [87] plied LSB and modulo three strategies to improve the performances. Although Wang et al.'s [84] MFPVD performs well, but gives range mismatch problem. Swain [88, 89] proposed some improvements. Various flavours of MF based techniques are provided by researchers [90-94, 110, 111].

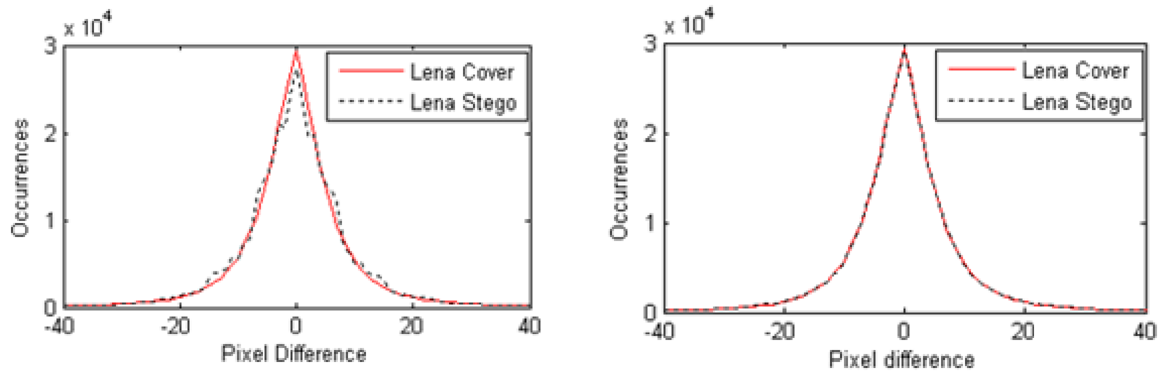
2.2.6 Edge Based Techniques

In edge regions the changes in pixel values cannot be easily noticed, so we can put good number of bits in its edge areas in contrast to smooth areas. The edges in the image are often selected by a variety of detectors like Canny, Fuzzy, and Sobel. Islam et al. [95] developed an edge-based scheme dependent on the message size. It successfully survives various attacks. Sun [96] proposed Canny edge detector-based technique to find the edges and effectively improved HC. Luo et al. [97] identified the horizontal and vertical edges as per difference between two consecutive pixels adaptively. Modi et al. [98] modified Canny edge detector to improve the performance, but could not give better HC as compared to others. Bassil et al. [99] brought one Canny edge related LSB approach to keep the secret bits in RGB channels. This approach buries secret bits in each channel, in their 3LSBs. Chan and Chang [100] combined both Canny and Fuzzy detector. Roy et al. [101] found the edges by matrix encoding strategy to enhance the HC. Dadgostar and Afsari [102] correctly found the edges by a new variant of fuzzy edge detector. In literature, different flavors of edge detectors are proposed to improve upon HC [103-106].

2.3. Security Evaluation

2.3.1 PDH Attack

The SIs of usual PVD schemes could be detected by PDH attack. The PDH is a 2-D graph, where pixel difference (PD) and frequency count of PD are taken on x and y axes accordingly [3]. We consider two plots together for comparison. One plot is for the designated OI and other plot for its SI. The curve of OI will be smooth in shape without curly appearance. If the curve of the SI is also smooth, then PDH test could not detect it. If the curve of the SI is also curly in nature, then PDH test could detect it. For examples, Fig.2.5 (a) renders the PDH test for Lena image in Wu and Tsai [49] technique, Fig.2.5 (b) renders PDH test for Lena image Luo et al.'s [9] scheme. In Fig.2.5 (a), we can notice curly nature in the curve of SI, so PDH test could detect it. In Fig.2.5 (b) there is no curly nature in the curve of SI, so PDH test could detect it.



(a)Wu and Tsai's PVD technique [49]

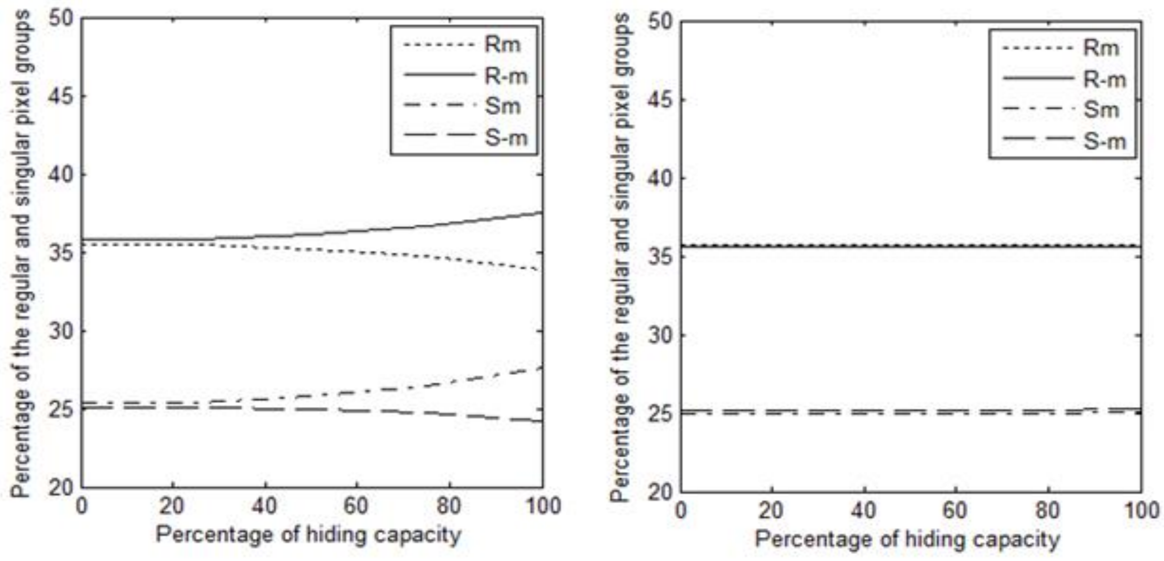
(b)Luo et al.'s APVD technique [9]

Fig.2.5 PDH attack examples

2.3.2 RS Attack

RS analysis is a statistical analysis. It uses the ideology that in LSB substitution, the change in pixel value 2n ↔2n+1 does happen, but the change 2n ↔2n-1 does not happen. RS analysis computes 4 parameters R_m , S_m , R_{-m} and S_{-m} as described in [3] and detects the steganography. In the OI the condition $R_m \approx R_{-m} > S_m \approx S_{-m}$ will be satisfied usually. If it also satisfies in SI, then RS test could not detect it.

"In contrary, if for SI, we get $R_{-m} - S_{-m} > R_m - S_m$, then steganography is detected. Fig.2.6(a) is an example of RS test over 1-bit LSB substitution scheme. Fig.2.6(b) is an example of RS test over Luo et al.'s [9] methodology. In Fig.2.6(b), $R_m \approx R_{-m} > S_m \approx S_{-m}$. Hence Luo et al.'s scheme is un-detected by RS test. In Fig.2.6(a), $R_{-m} - S_{-m} > R_m - S_m$, so 1-LSB substitution is detected by RS test" [3].



(a) 1-bit LSB substitution technique (b) Adaptive PVD technique [10]

Fig.2.6 RS attack over Lena image

2.4. Identified Research Problems

This research focusses on three research issues, (i) FOBP in recent PVD based techniques, (ii) hiding capacity of APVD techniques are low due to UBP, and (iii) PDH analysis does not detect MDPVD techniques.

2.4.1 FOBP in PVD Steganography

For any gray image, its pixel value is from 0 to 255. After embedding the secret bits in it, if its value does not fall in {0, 255}, then FOBP has occurred.

In Khodaei and Faez's [68] scheme, FOBP occurs. Jung's [73] LSB+PVD also suffers from FOBP. We give below a numerical argument to understand it. Say, k=2, and a block $(P_1, P_2) = (129, 249)$. That means P_1 = 129 and P_2= 249. We can compute $R_1 = 1, R_2 = 1, Q_1 = 32$, and $Q_2 = 62$. We want to hide the data 00010010_2. Take 2 bits i.e., 00_2, so R'_1 = 0. Again, take next 2 bits i.e., 01_2, so R'_2 =1. As $Q_1 = 32$, and $Q_2 = 62$, so d= |32 - 62| = 30. As our d value 30 falls in {16, 31}, so we got L=16 and n=4. Again, the next 4 bits are 0010_2, i.e., S=2. Further we computed d' =L+S = 16 + 2 = 18, m=$|d' - d| = |18 - 30| = 12$. Here d is an even value. So, we got $Q'_1 = \left(Q_1 - \left\lfloor \frac{m}{2} \right\rfloor\right) = 26$, and $Q'_2 = \left(Q_2 + \left\lceil \frac{m}{2} \right\rceil\right) = 68$. Finally, the stego-pixel values, P'_1 = 26×4+0 = 104 and P'_2 = 68×4+1 = 273. The value of P'_2 > 255, so FOBP identified.

2.4.2 Unused Blocks Problem (UBP) in APVD

APVD scheme of Luo et al.'s [9] has been described in this Chapter. Table 2.7 shows a count of the blocks matching with the condition ($|d_1| \leq$ T and $|d_2| \leq$ T), for T value as 10. It is evident from this study that 69 % of the blocks are unassessed. As a consequence of this, the HC falls down to 0.37 BPB.

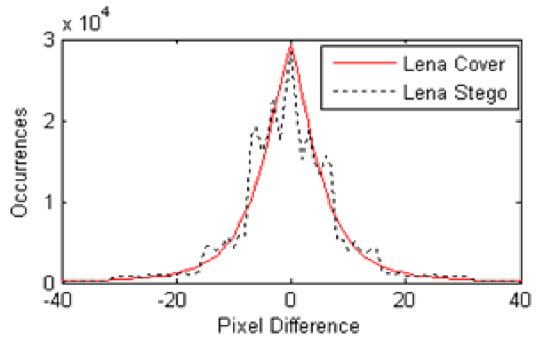
Table 2.7 Un-used blocks in Luo et al.'s [9] APVD technique to hide 140,000 bits of data

Images 512×512×3	Number of 1×3 size blocks satisfied the condition ($\lvert d_1\rvert \leq$ T and $\lvert d_2\rvert \leq$ T)	Total number of 1×3 size blocks accessed to hide 140,000 bits of data	% of 1×3 size unused blocks satisfied the condition ($\lvert d_1\rvert \leq$ T and $\lvert d_2\rvert \leq$ T)
Lena	124546	172790	72.08
Baboon	6841	47436	14.42
Tiffany	170420	218990	77.82
Peppers	90868	139176	65.29
Jet	199332	245361	81.24
Boat	57906	102881	56.28
House	71114	115386	61.63
Average	103003	148860	69.20

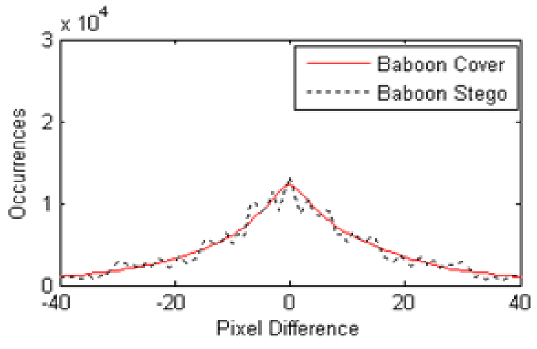
2.4.3 PDH Analysis cannot Detect Multi-directional PVD

The conventional PVD schemes are caught by PDH test. For an examples, Fig.2.7 (a) and (b) depicts PDH tests on Lena and Baboon images for Wu and Tsai's PVD. In both the cases the continuous line depicts the OI and dot line depicts the SI. The dot line is curly in nature, so PDH detected the steganography.

Researchers found that by using large blocks like 2×3 and 3×3, and exploiting edges in multiple directions, PDH test cannot detect. The traditional PDH test can catch only those schemes that plies PVD in 1×2 blocks. But it cannot trace steganography schemes that uses PVD in larger size blocks like 4, 6, and 9.



(a)



(b)

Fig.2.7 PDH analysis examples, (a) for Lena image, and (b) for Baboon image.

2.5 Research Contributions

To address these three problems stated above, three different steganography techniques are proposed in Chapters 3, 4, and 5. In Chapter 3, a steganography technique based on QVD and PVC is proposed. It avoids FOBP. In Chapter 4, a steganography technique based on RR, AQVD, and QVC is proposed. It makes a fair balance between PSNR, and HC. It also addresses UBP. In Chapter 5, the MDPDH analysis is proposed, which can detect multi-directional PVD techniques.