Chapter 5

Steganography Using Pixel Value Differencing

5.1 Introduction

In this Chapter two techniques based on PVD are described. The first technique is known as two sided, three sided and four sided side match steganography. The second technique is known as five, six, seven and eight neighbor correlation steganography. In both these techniques the number of bits to be embedded in a target pixel is decided based on the difference of target pixels’ value and the average value of its neighboring pixels. Both these techniques represent variable bit rate embedding.

The first technique is discussed in section 5.2 and the second technique is discussed in section 5.3 in detail. A comparison study is presented in section 5.4, and the Chapter is concluded in section 5.5.

5.2 Steganography Using Two Sided, Three Sided, and Four Sided Side Match Methods

In this section the two sided, three sided and four sided side match methods are discussed. In two sided side match method the two neighboring pixels such as upper and upper-right corner are exploited to take embedding decision. In three sided side match method the three neighboring pixels such as upper, upper-right corner and upper-left corner are exploited to take embedding
decision. In four sided side match method the four neighboring pixels such as upper, left, upper-right corner and upper-left corner are exploited to take embedding decision. In each of these methods the fall off boundary problem (FOBP) and fall in error problem (FIEP) are addressed. The FOBP and FIEP conditions are to be checked while embedding at the sender and also while extracting at the receiver. These methods possess high embedding capacity, better imperceptibility and security.

5.2.1 The Proposed Side Match Methods

The two sided, three sided and four sided side match methods proposed by Chang and Tseng [31] do not address the FIEP. The two sided, three sided and four sided side match methods proposed in this paper addresses both FIEP and FOBP problems. These are discussed in the following sub-sections.

5.2.1.1 Two Sided Side Match Steganography

The cover image is scanned in the raster-scan order. The pixels are categorized into three categories as shown in Fig.5.1. The pixels shaded with gray color are used for embedding and also as neighbors. But the white colored pixels are used as neighbors only. The pixels shaded with light gray color are neither used for embedding nor as neighbors.

The two sided side match method uses the upper and upper-right neighbor pixels for estimating the number of bits to be embedded in a target pixel. Suppose $P_{x}$ is a target pixel with gray value $g_{x}$, $P_{U}$ is its upper neighboring pixel with gray value $g_{u}$ and $P_{UR}$ is its upper-right neighboring pixel with gray value $g_{ur}$. Then to estimate the number of bits that can be embedded in the target pixel, a difference value $d$ is computed as in equation 5.1.

$$d = \frac{g_{u} + g_{ur}}{2} - g_{x}$$  \hspace{1cm} (5.1)

If the pixels are in a smooth area we will get a smaller value of $d$ and if the pixels are in an edge area we will get a larger value of $d$. Larger the value of $d$ means more number of bits can be embedded. If $d$ has a value -1, 0, or 1 then the pixel is not used for embedding. If $d$ has a value greater than or equal to 2 then we can hide $n$ bits in that pixel. The value $n$ is calculated as in equation 5.2.
Now a sub-stream of $n$ bits of secret data is taken, converted to integer value $b$ and the new difference value $d'$ is computed as in equation 5.3.

$$d' = \begin{cases} 2^n + b, & \text{if } d > 1 \\ -(2^n - b), & \text{if } d < 1 \end{cases}$$  \hspace{1cm} (5.3)$$

Hence the new value of the pixel after embedding $n$ secret bits is computed as in equation 5.4.

$$g_x' = \frac{(g_u + g_{ur})}{2} - d'$$  \hspace{1cm} (5.4)$$

The extraction procedure is also very simple. The embedded data is extracted in raster-scan order. Suppose $P^*_x$ is the target pixel with gray value $g^*_x$. $P^*_u$ is its upper neighboring pixel with gray value $g^*_u$ and $P^*_{ur}$ is its upper-right neighboring pixel with gray value $g^*_{ur}$. Then to estimate the number of bits embedded, a difference value $d^*$ is computed as in equation 5.5.

$$d^* = \frac{(g^*_u + g^*_{ur})}{2} - g^*_x$$  \hspace{1cm} (5.5)$$

If $d^*$ has a value -1, 0 or 1, then that pixel is ignored. Otherwise the number of bits, say $n$, which was embedded, is calculated by equation 5.6.

$$n = \log_2 |d^*|, \text{ if } |d^*| > 1$$  \hspace{1cm} (5.6)$$
Now the integer equivalent of embedded bit stream, \( b \) is computed as in equation 5.7.

\[
b = \begin{cases} 
    d' - 2^n, & \text{if } d' > 1 \\
    -d' - 2^n, & \text{if } d' < 1 
\end{cases}
\]  

(5.7)

This integer value \( b \) is then converted to \( n \) bit binary.

Sometimes the new value of pixel \( P_X \) may fall off boundary of the range \( \{0, 255\} \) in the following cases as discussed by Chang and Tseng [31].

Case 1: \( d > 1 \) and \( (g_u + g_{ur}) / 2 < 2^{n+1} - 1 \)

From equation 5.3 and 5.4, \( g_x' = (g_u + g_{ur}) / 2 - d = (g_u + g_{ur}) / 2 - (2^n + b) \).

Assume that the embedding value \( b \) is given by the largest value \( 2^n - 1 \), then \( g_x' = (g_u + g_{ur}) / 2 - (2^n + 2^n - 1) = (g_u + g_{ur}) / 2 - (2^{n+1} - 1) \).

Therefore, \( g_x' \) will be a negative quantity when \( (g_u + g_{ur}) / 2 < (2^{n+1} - 1) \).

Case 2: \( d < 1 \) and \( (g_u + g_{ur}) / 2 + 2^{n+1} > 256 \)

From equation 5.3 and 5.4, \( g_x' = (g_u + g_{ur}) / 2 - d = (g_u + g_{ur}) / 2 + (2^n + b) \).

Assume that the embedding value \( b \) is given by the largest value \( 2^n - 1 \), then \( g_x' = (g_u + g_{ur}) / 2 + (2^n + 2^n - 1) = (g_u + g_{ur}) / 2 + 2^{n+1} - 1 \).

Therefore, \( g_x' \) will be greater than 255 when \( (g_u + g_{ur}) / 2 + 2^{n+1} > 256 \).

Suppose for \( d = -1, 0 \) or 1 only one bit data will be embedded using traditional LSB method as it is done by Chang and Tseng [31]. An error condition occurs for \( d = -1 \), the extraction at receiver fails. Let us see through an example. Suppose in two neighbor case \( g_x = 32, g_u = 31 \) and \( g_{ur} = 31 \). Then \( d = (31 + 31)/2 - 32 = -1 \). The value of \( g_x = 32 \) in binary it is 00100000. For \( d = -1 \), only one bit data is to be embedded by using LSB substitution. Suppose the one bit to be embedded is bit 1, then the new value of \( g_x \) after embedding is 00100001 = 33. Now at receiver \( g_x' = 33, g_u' = 31 \) and \( g_{ur}' = 31 \). d* = \( (31 + 31) / 2 - 33 = -2 \). As \( d^* \) at receiver is not 1, 0 or -1, so by applying equation 5.6 we get, \( n = \log_2 |d^*| = 1 \). Next by using equation 5.7, we get \( b = -d^* - 2^1 = 2 - 2 = 0 \). The bit extracted is 0, but the embedded bit is 1. Thus this is an error condition. It is better to include the conditions \( d = 0 \) and \( d = 1 \) too for the sake of uniformity. So a pixel where \( d = -1, 0 \) or 1 is said to fall in error problem. This is considered as case 3.
Case 3: If \( d = -1, 0 \) or 1 then do not embed in that pixel

The falling off boundary checking and the falling in error checking is applied at the sender for data embedding and at the receiver for data extraction. The pixel \( P_X \) which falls in these three conditions (i.e. case 1, case 2 and case 3) is not used for embedding at the sender. Similarly data is not extracted from it at the receiver.

Let us discuss an example for embedding and extraction. Suppose the pixel \( P_X \) has a gray value \( g_X = 45 \), its upper neighbor pixel \( P_U \) has a gray value \( g_u = 40 \) and its upper-right neighbor pixel \( P_{UR} \) has a gray value \( g_{ur} = 42 \).

Then \( d = (40+42)/2 - 45 = -4 \) as per equation 5.1.

The number of bits, \( n \) that can be embedded in pixel \( P_X \) is 2 as per equation 5.2. Suppose the two binary bits to be embedded are \((10)_2\). Its integer value is 2, i.e. \( b = 2 \). Then \( d' = -6 \) as per equation 5.3 and the new value of pixel after embedding, \( g'_X = (g_u + g_{ur})/2 - d' = 41 - (-6) = 47 \).

At receiver the pixel \( P'_X \) has gray value \( g'_X = 47 \), its upper neighbor \( P'_U \) has gray value, \( g'_u = 40 \) and upper-right neighbor \( P'_{UR} \) has gray value, \( g'_{ur} = 42 \).

Then the difference \( d^* = (g'_u + g'_{ur})/2 - g'_X = 41 - 47 = -6 \) as per equation 5.5. The number of bits that to be extracted is, \( n = \log_2 | -6 | = 2 \) (the lower rounded integer value is taken). Thus \( b \) value is 2 as per equation 5.7. In two bit binary the integer 2 is represented as \((10)_2\), which are the two bits embedded at the sender.

### 5.2.1.2 Three Sided Side Match Steganography

The cover image is scanned in the raster-scan order. The pixels are categorized into three categories as shown in Fig.5.2. The pixels shaded with gray color are used for embedding and as neighbors. The white colored pixels are used as neighbors only. The pixels shaded with light gray color are neither used for embedding nor as neighbors.

The three sided side match method uses the upper, upper-right and upper-left neighbor pixels for estimating the number of bits to be embedded in a target pixel. Suppose \( P_X \) is a target pixel with gray value \( g_X \), \( P_U \) is its upper neighboring pixel with gray value \( g_u \), \( P_{UR} \) is its upper-right neighboring pixel with gray value \( g_{ur} \), and \( P_{UL} \) is its upper-left neighboring pixel with gray value \( g_{ul} \).
Then to estimate the number of bits to be embedded, a difference value $d$ is computed as in equation 5.8.

$$d = (g_u + g_{ur} + g_{ul}) / 3 - g_x$$  \hspace{1cm} (5.8)

If $d$ has a value -1, 0, or 1, then in that pixel no data is embedded. If $d$ has a value greater than or equal to 2, then we can hide $n$ bits in that pixel. The value $n$ is calculated as in equation 5.9.

$$n = \log_2 |d|, \quad \text{if } |d| > 1$$  \hspace{1cm} (5.9)

Now a sub-stream of $n$ bits of secret data is taken, converted to integer value $b$ and the new difference value $d'$ is computed as in equation 5.10.

$$d' = \begin{cases} 2^n + b, & \text{if } d > 1 \\ -(2^n + b), & \text{if } d < 1 \end{cases}$$  \hspace{1cm} (5.10)

Hence the new value of the pixel after embedding $n$ secret bits is computed as in equation 5.11.

$$g'_x = (g_u + g_{ur} + g_{ul}) / 3 - d'$$  \hspace{1cm} (5.11)

The extraction procedure is identical to that of two sided side match method. The FOBP and FIEP conditions are also taken care as in two sided side match method.
5.2.1.3 Four Sided Side Match Steganography

The cover image is scanned in the raster-scan order. The pixels are categorized into three categories as shown in Fig.5.3. The pixels shaded with gray color are used for embedding and as neighbors. The white colored pixels are used as neighbors only. The pixel shaded with light gray color is neither used for embedding nor as neighbor.

![Types of pixels in four sided side match method](image)

The four sided side match method uses the upper, upper-right, upper-left and left neighbor pixels for estimating the number of bits to be embedded in a target pixel. Suppose $P_X$ is a target pixel with gray value $g_X$, $P_U$ is its upper neighboring pixel with gray value $g_U$, $P_{UR}$ is its upper-right neighboring pixel with gray value $g_{UR}$, $P_{UL}$ is its upper-left neighboring pixel with gray value $g_{UL}$ and $P_L$ is its left neighboring pixel with gray value $g_L$. Then to estimate the number of bits to be embedded, a difference value $d$ is computed as in equation 5.12.

$$d = \frac{(g_U + g_{UR} + g_{UL} + g_L)}{4} - g_X$$  \hspace{1cm} (5.12)

If $d$ has a value -1, 0, or 1, then in that pixel no data is embedded. If $d$ has a value greater than or equal to 2, then we can hide $n$ bits in that pixel. The value $n$ is calculated as in equation 5.13.

$$n = \log_2 |d|, \text{ if } |d| > 1$$  \hspace{1cm} (5.13)

Now a sub-stream of $n$ bits of secret data is taken, converted to integer value $b$ and the new difference value $d'$ is computed as in equation 5.14.
\[ d' = \begin{cases} 
2^n + b, & \text{if } d > 1 \\
-(2^n + b), & \text{if } d < 1 
\end{cases} \quad (5.14) \]

Hence the new value of the pixel after embedding \( n \) secret bits is computed as in equation 5.15.

\[ g'_x = \left( g_u + g_{ur} + g_{ul} + g_l \right) / 4 - d' \quad (5.15) \]

The extraction procedure is identical to that of two sided side match method. The FOBP and FIEP conditions are also taken care as in two sided side match method.

### 5.2.2. Experimental Results and Discussion

The methods are implemented using MATLAB and are tested with various images. The observations for four standard images are as discussed below.

Fig.5.4 (a) is the Airplane image with a size of 192 kilo bytes. Fig.5.4 (b)-(d) are the stego-images in two sided, three sided and four sided side match methods respectively with 10240 bytes of data hidden in each. Fig.5.4 (e)-(h) are their histograms.

Fig.5.5 is the Lena image with a size of 535 kilo bytes. Fig.5.5 (b)-(d) are the stego-images in two sided, three sided and four sided side match methods with 20480 bytes of data hidden in each. Fig.5.5 (e)-(h) are their histograms.

Fig.5.6 is the Baboon image with a size of 525 kilo bytes. Fig.5.6 (b)-(d) are the stego-images in two sided, three sided and four sided side match methods respectively with 12288 bytes of data hidden in each. Fig.5.6 (e)-(h) are their histograms.
Fig. 5.4 (a) Airplane image; (b), (c), (d) stego-images; (e), (f), (g), (h) histograms
Fig. 5.5 (a) Lena image; (b), (c), (d) stego-images; (e), (f), (g), (h) histograms
Fig. 5.6 (a) Baboon image; (b), (c), (d) stego-images; (e), (f), (g), (h) histograms
The performance of various steganography algorithms can be rated by the three parameters; (i) security, (ii) capacity, and (ii) imperceptibility. These proposed steganography methods are secure because instead of replacing the LSBs of pixel values directly, the proposed methods adaptively change the pixel value into another value based on the values of the neighboring pixels. The traditional LSB steganography have a common weak point. The pixel value changes asymmetrically. When the LSB of cover medium pixel value is equal to the message bit, no change is made. Otherwise the value is changed from $2n$ to $2n+1$ or $2n+1$ to $2n$. But the change from $2n$ to $2n-1$ or $2n+1$ to $2n+2$ never happens. This asymmetry can be captured by RS steganalysis. The methods proposed in this paper are not vulnerable to steganalysis methods like RS steganalysis because we are not replacing the LSBs rather we are changing the pixel values. As the histograms of original images and stego-images are completely identical, histogram based steganalysis can not detect this steganography.

Capacity means the amount of message that can be hidden. To be useful in conveying secret message, the hiding capacity provided by steganography should be as high as possible, which may be given in absolute measurement such as the size of secret message, or in relative value called data embedding rate, such as bits per pixel, or the ratio of the secret message to the cover medium. In two, three and four sided side match methods we are hiding data in all the pixels except the first column, first row and last column of the image. As per the sampling arrangement of the pixels in Chang and Tseng’s [31] three and four sided side match methods only 50% of the pixels can be targeted, but in our methods except the pixels of first row, first column and last column all the remaining pixels can be targeted. Thus our methods can have higher embedding capacity.

Stego-images should not have severe visual artifacts. Without sacrificing the level of security and hiding capacity, if the imperceptibility is higher, then the steganography technique is better. If the resultant stego-image appears innocuous enough, then one can say that this requirement is satisfied. It can be observed from Fig.5.4-5.6 that the stego-images look very innocuous. We can measure the distortion in the stego-image by PSNR value. The higher is the PSNR value means less is the distortion. The PSNR values for various images in these three methods are as shown in Table 5.1. With the same amount of hidden data the distortion is the minimum in four sided side match method as compared to two and three sided side match methods. The correlation, which is
a measure of similarity between the cover image and stego-image, recorded in Table 5.2 is acceptable. If the cover image and stego images are very similar then the correlation value will be close to 1. The K-L divergence between the cover image and stego-images is as shown in Table 5.3, these are also acceptable.

Table 5.1 PSNR values of proposed two, three and four sided side match methods

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Amount of hidden data (bytes)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Two sided</td>
<td>Three sided</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>10240</td>
<td>39.8040</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>20480</td>
<td>44.8240</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>12288</td>
<td>40.8951</td>
</tr>
</tbody>
</table>

Table 5.2 Correlation values of proposed two, three and four sided side match methods

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Amount of hidden data (bytes)</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Two sided</td>
<td>Three sided</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>10240</td>
<td>0.9983</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>20480</td>
<td>0.9997</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>12288</td>
<td>0.9992</td>
</tr>
</tbody>
</table>

Table 5.3 K-L Divergence values of proposed two, three and four sided side match methods

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Amount of hidden data (bytes)</th>
<th>K-L Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Two sided</td>
<td>Three sided</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>10240</td>
<td>91.81</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>20480</td>
<td>40.7</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>12288</td>
<td>13.1</td>
</tr>
</tbody>
</table>

The embedding capacity of the proposed two, three and four sided side match methods are compared with the two, three, and four sided side match methods of Chang and Tseng, in Table 5.4. In our two and three sided methods the capacity is almost doubled as compared to Chang and Tseng’s methods. And in four sided side match method the capacity is almost four times improved.
Table 5.4 Comparison of capacity with Chang and Tseng’s method

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>proposed methods</th>
<th>Chang and Tseng’s methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Two sided</td>
<td>Three sided</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>317528</td>
<td>314693</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>710538</td>
<td>653012</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>1555767</td>
<td>1516036</td>
</tr>
</tbody>
</table>

Fig. 5.7 (a)-(c) represents the variation of PSNR value with payload for Airplane image. From the graph we can observe that if we go on increasing the payload up to a certain maximum allowable value, the PSNR may not reach less than 40.
Fig. 5.7 Variation of PSNR with payload for Airplane image in (a) two-sided, (b) three-sided, and (c) four-sided methods.
5.3 Steganography Using the Correlation of Five, Six, Seven and Eight Neighbor Pixels

In this section five, six, seven and eight neighbor correlation methods are discussed. In five neighbor correlation method the five neighboring pixels such as upper, left, right, upper-left corner and upper-right corner are exploited to take embedding decision. In six neighbor correlation method the six neighboring pixels such as upper, left, right, upper-left corner, upper-right corner and bottom-left corner are exploited to take embedding decision. In seven neighbor correlation method the seven neighboring pixels such as upper, left, right, bottom, upper-left corner, upper-right corner and bottom-left corner are exploited to take embedding decision. In eight neighbor correlation method all the eight neighboring pixels are exploited to take embedding decision. In each of these methods the FOBP and FIEP are addressed. The FOBP and FIEP conditions are to be checked while embedding at the sender and while extracting at the receiver. These methods possess high embedding capacity, better imperceptibility and security.

5.3.1 The Proposed Methods Using Correlation with Neighbor Pixels

The five, six, seven and eight neighbor correlation methods are discussed in following subsections. The FOBP and FIEP are also addressed.

5.3.1.1 Five Neighbor Correlation Steganography

The cover image is scanned in raster-scan order. The pixels are categorized into two categories as shown in Fig.5.8. The pixels those are marked with deep green color are used for embedding data and as neighbors for their bottom pixels. The white pixels are used as neighbors only. The pixels marked with deep green color in the last row are used for embedding only.

The values of the upper neighboring pixel \( P_U \), left neighboring pixel \( P_L \), right neighboring pixel \( P_R \), upper-left corner pixel \( P_{UL} \) and upper-right corner pixel \( P_{UR} \) are used to estimate the number of bits to be embedded in a target pixel. Given a target pixel \( P_X \) with gray value \( g_X \); let \( g_u, g_l, g_r, g_{ul} \) and \( g_{ur} \) be the gray values of its upper \( P_U \), left \( P_L \), right \( P_R \), upper-left \( P_{UL} \) and upper-right \( P_{UR} \) respectively. Then a difference value \( d \) is computed as in equation 5.16.
Steganography Using Pixel Value Differencing

\[ d = \frac{(g_u + g_l + g_r + g_{ul} + g_{ur})}{5} - g_x \]  
(5.16)

A small difference value indicates that the pixel is in a smooth area, whereas a large difference value indicates that the pixel is in an edge area. In the pixels of edge areas more number of bits can be embedded compared to pixels in smooth areas. If \( d \) has a value -1, 0, or 1 then the pixel is not used for embedding data in it. If \( d \) has a value greater than or equal to 2 then we can hide \( n \) bits in that pixel. The value \( n \) is calculated as in equation 5.17.

\[ n = \log_2 |d|, \quad \text{if } |d| > 1 \]  
(5.17)

A sub-stream of \( n \) bits from the secret data is taken and is converted to integer \( b \). Then a new difference \( d' \) is computed as in equation 5.18.

\[ d' = \begin{cases} 2^n + b, & \text{if } d > 1 \\ -(2^n + b), & \text{if } d < 1 \end{cases} \]  
(5.18)

Finally the new value of the pixel \( P_X \) after embedding is as defined in equation (5.19).

\[ g'_{X} = \frac{(g_u + g_l + g_r + g_{ul} + g_{ur})}{5} - d' \]  
(5.19)

The extraction procedure is also very simple. The embedded data is extracted in raster-scan order. Given a target pixel \( P_X^* \) with gray value \( g^*_X \); let \( g_u^*, g_l^*, g_r^*, g_{ul}^*, g_{ur}^* \) and \( g_{ul}^* \) and \( g_{ur}^* \) be the gray values
of the upper pixel $P^*_U$, left pixel $P^*_L$, right pixel $P^*_R$, upper-left pixel $P^*_{UL}$ and upper-right pixel $P^*_{UR}$ respectively. Then a difference value $d^*$ is computed as in equation 5.20.

$$d^* = (g^*_u + g^*_l + g^*_r + g^*_{ul} + g^*_{ur}) / 5 - g^*_x.$$  \hspace{1cm} (5.20)

If $d^*$ has a value -1, 0 or 1, then that pixel is ignored. Otherwise the number of bits (say n) which was embedded is calculated by equation 5.21.

$$n = \log_2 |d^*|, \text{ if } |d^*| > 1$$ \hspace{1cm} (5.21)

Finally, the embedded value $b$ is extracted out using the equation 5.22.

$$b = \begin{cases} d' - 2^n, & \text{if } d' > 1 \\ -d' - 2^n, & \text{if } d' < 1 \end{cases}$$ \hspace{1cm} (5.22)

The value $b$ is then converted to its corresponding binary in n bit length. Sometimes, the new value of pixel $P_X$ may fall off the boundary of the range \{0, 255\} in the following cases as discussed by Chang and Tseng [31].

**Case 1:** $d > 1$ and $(g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 < 2^{n+1} - 1$

From equation 5.18 and 5.19, $g'_x = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 - d' = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 - (2^n + b)$. Assume that the embedding value $b$ is given by the largest value $2^n - 1$, then,

$$g'_x = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 - (2^n + b) = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 - (2^{n+1} - 1).$$

Therefore, $g'_x$ will be a negative quantity when $(g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 < (2^{n+1} - 1)$.

**Case 2:** $d < 1$ and $(g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 + 2^{n+1} > 256$

From equation 5.18 and 5.19, $g'_x = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 - d' = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 + (2^n + b)$. Assume that the embedding value $b$ is given by the largest value $2^n - 1$, then,

$$g'_x = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 + (2^n + b) = (g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 + 2^{n+1} - 1.$$ Therefore, $g'_x$ will be greater than 255 when $(g_u + g_l + g_r + g_{ul} + g_{ur}) / 5 + 2^{n+1} > 256.$
Suppose for $d = -1, 0$ or $1$ only one bit data is to be embedded using traditional LSB method because equation 5.17 can not be applied as told by Chang and Tseng [31].

An error condition occurs for $d = -1$, the extraction at receiver fails. As an example suppose in five neighbor case $g_x = 32$, $g_u = 31$, $g_l = 31$, $g_r = 31$, $g_{ul} = 31$ and $g_{ur} = 31$. Then $d = (31+31+31+31+31)/5 - 32 = -1$. The value of $g_x = 32$ in binary it is 00100000. For $d = -1$, only one bit data is to be embedded by using LSB substitution method. Suppose the one bit to be embedded is bit 1, then the new value of $g_x$ after embedding is 00100001 = 33. Now at receiver $g_x = 33$, $g_{ul} = 31$, $g_{ur} = 31$, $g_r = 31$, $g_{ul} = 31$ and $g_{ur} = 31$, $d' = (31 + 31 + 31 + 31 + 31)/5 - 33 = -2$. As $d' = 1$, 0 or -1, so by applying equation 5.21 we get, $n = \log_2 |d'| = 1$, Next by using equation 5.22, we get $b = -d' - 2^1 = 2 - 2 = 0$, the bit extracted is 0, but the embedded bit is 1. Thus this is an error condition. It is better to include the conditions $d = 0$ and $d = 1$ too for the sake of uniformity. So a pixel where $d = -1, 0$ or 1 is said to fall in error problem. This is treated as case 3.

Case 3: If $d = -1, 0$ or 1, then do not embed in that pixel

The falling off boundary checking and the falling in error checking is applied to data embedding and extraction. The pixel $P_X$ which falls in these three conditions (i.e. case 1, case 2, and case 3) is not embedded at the sender. Similarly data is not extracted from it at the receiver.

Let us discuss an example for embedding and extraction. Suppose the pixel $P_X$ has a gray value $g_x = 45$, its five neighbors have gray values 40, 40, 41, 42 and 42.

Then $d = (40+40+41+42+42)/5 - 45 = -4$ as per equation 5.16.

The number of bits $n$ that can be embedded in pixel $P_X$ is 2 as per equation 5.17. Suppose the two binary bits to be embedded are $10$. Its integer value is 2 i.e. $b = 2$. Then $d' = -6$ as per equation 5.18 and the new value of pixel after embedding, $g_x' = (40+40+41+42+42)/5 - (-6) = 47$ as per equation 5.19.

At receiver the pixel $P_X$ has gray value $g_x' = 47$, its neighbors have gray values 40, 40, 41, 42 and 42. Then the difference $d^* = (40 + 40 + 41 + 42 + 42)/5 - 47 = -6$ as per equation 5.20. The number of bits to be extracted is, $n = \log_2 |-6| = 2$ (the lower integer value is taken). Thus $b$ value is 2 as per equation 5.22. In two bit length the integer value 2 is represented as $10$ in binary, which are the two bits embedded at the sender.
5.3.1.2 Six neighbor correlation Steganography

The cover image is scanned in the raster-scan order. The pixels are categorized into three categories as shown in Fig. 5.9. The pixels marked with deep green color are used for embedding data and as neighbors for their bottom pixels. The white pixels are used as neighbors only. The pixels marked with light green are neither used for embedding nor as neighbors.

The six neighbor correlation method uses the upper, left, right, upper-right, upper-left and bottom-left neighbor pixels for estimating the number of bits to be embedded in a target pixel. Let \( g_x \) be the gray value of a target pixel \( P_X \); \( g_u, g_l, g_r, g_{ur}, g_{ul} \) and \( g_{bl} \) be the gray values of its upper pixel \( P_U \), left pixel \( P_L \), right pixel \( P_R \), upper-right pixel \( P_{UR} \) and upper-left pixel \( P_{UL} \) and bottom-left pixel \( P_{BL} \) respectively. Then the difference value \( d \) is computed as in equation 5.23.

\[
    d = \frac{(g_u + g_l + g_r + g_{ur} + g_{ul} + g_{bl})}{6} - g_x
\]  

(5.23)

If \( d \) has a value -1, 0, or 1 then the pixel is not used for embedding data in it. If \( d \) has a value greater than or equal to 2, then we can hide \( n \) bits in that pixel. The value \( n \) is calculated as in equation 5.24.

\[
    n = \log_2 |d|, \quad \text{if } |d| > 1
\]  

(5.24)

![Diagram of pixel types in six neighbor correlation method](image)

Fig. 5.9 Types of pixels in six neighbor correlation method
A sub-stream with \( n \) bits of secret data is taken and is converted to integer \( b \). Then a new difference \( d' \) is computed as in equation 5.25.

\[
d' = \begin{cases} 
2^n + b, & \text{if } d > 1 \\
-(2^n + b), & \text{if } d < 1 
\end{cases}
\]  

(5.25)

And the new value of the pixel \( P_X \) is defined as in equation 5.26

\[
g_X' = \frac{(g_u + g_l + g_r + g_{ur} + g_{ul} + g_{bl})}{6} - d' 
\]

(5.26)

The extraction procedure is identical to that of five neighbor correlation method. The FOBP and FIEP conditions are also taken care as in five neighbor correlation method.

### 5.3.1.3 Seven Neighbor Correlation Steganography

The cover image is scanned in the raster-scan order. The pixels are categorized into three categories as shown in Fig. 5.10. The pixels marked with deep green color are used for embedding. The white pixels are used as neighbors only. The pixel marked with light green color are neither treated as neighbors nor to embed data.

![Diagram of seven neighbor correlation method](image-url)

Fig.5.10 Types of pixels in seven neighbor correlation method
The seven neighbor correlation method uses the upper, left, right, bottom, upper-right, upper-left and bottom-left neighbor pixels for estimating the number of bits to be embedded in a target pixel. Let \( g_x \) be the gray value of a target pixel \( P_X \); \( g_u, g_l, g_r, g_b, g_{ur}, g_{ul}, \) and \( g_{bl} \) be the gray values of its upper pixel \( P_U \), left pixel \( P_L \), right pixel \( P_R \), bottom pixel \( P_B \), upper-right pixel \( P_{UR} \), upper-left pixel \( P_{UL} \) and bottom-left pixel \( P_{BL} \) respectively. Then the difference value \( d \) is computed as in equation 5.27.

\[
d = \frac{(g_u + g_l + g_r + g_b + g_{ur} + g_{ul} + g_{bl})}{7} - g_x
\]  

(5.27)

If \( d \) has a value -1, 0, or 1 then the pixel is not used for embedding data in it. If \( d \) has a value greater than or equal to 2 then we can hide \( n \) bits in that pixel. The value \( n \) is calculated as in equation 5.28.

\[
n = \log_2 |d|, \quad \text{if } |d| > 1
\]  

(5.28)

A sub-stream of \( n \) bits from secret data is taken and is converted to integer \( b \). Then a new difference value \( d' \) is computed as in equation 5.29.

\[
d' = \begin{cases} \
2^n + b, & \text{if } d > 1 \\
-(2^n + b), & \text{if } d < 1
\end{cases}
\]  

(5.29)

And the new value of the pixel \( P_X \) is defined as in equation 5.30.

\[
g'_x = \frac{(g_u + g_l + g_r + g_b + g_{ur} + g_{ul} + g_{bl})}{7} - d'
\]  

(5.30)

The extraction procedure is identical to that of five neighbor correlation method. The FOBP and FIEP conditions are also taken care as in five neighbor correlation method.

### 5.3.1.4 Eight Neighbor Correlation Steganography

The cover image is scanned in the raster-scan order. The pixels are categorized into two categories as shown in Fig.5.11. The pixels marked with deep green color are used for embedding data. The white pixels are used as neighbors only. This method uses the upper, left, right, bottom, upper-right, upper-left, bottom-left and bottom-right neighbor pixels for estimating the number of bits to be embedded in a target pixel. Let \( g_x \) be the gray value of a target pixel \( P_X \); \( g_u, g_l, g_r, g_b, g_{ur}, g_{ul}, g_{bl} \) and \( g_{br} \) be the gray values of its upper pixel \( P_U \), left pixel \( P_L \), right
Steganography Using Pixel Value Differencing

pixel \(P_R\), bottom pixel \(P_B\), upper-right pixel \(P_{UR}\), upper-left pixel \(P_{UL}\), bottom-left pixel \(P_{BL}\) and bottom-right pixel \(P_{BR}\) respectively. Then the difference value \(d\) is computed as in equation 5.31.

\[
d = \frac{(g_u + g_l + g_r + g_b + g_{ur} + g_{ul} + g_{bl} + g_{br})}{8} - g_x
\]  

(5.31)

If \(d\) has a value -1, 0, or 1 then the pixel is not used for embedding data in it. If \(d\) has a value greater than or equal to 2, then we can hide \(n\) bits in that pixel. The value \(n\) is calculated as in equation 5.32.

\[
n = \log_2 |d|, \quad \text{if} \quad |d| > 1
\]  

(5.32)

A sub-stream of \(n\) bits from secret data is taken and is converted to integer \(b\). Then a new difference value \(d'\) is computed as in equation 5.33.

\[
d' = \begin{cases} 
2^n + b, & \text{if } d > 1 \\
-(2^n + b), & \text{if } d < 1 
\end{cases}
\]  

(5.33)

And the new value of the pixel \(P_X\) is defined as in equation 5.34.

\[
g'_{X} = \frac{(g_u + g_l + g_r + g_b + g_{ur} + g_{ul} + g_{bl} + g_{br})}{8} - d'
\]  

(5.34)
The extraction procedure is identical to that of five neighbor correlation method. The FOBP and FIEP conditions are also taken care as in five neighbor correlation method.

### 5.3.2 Experimental Results and Discussion

The methods are implemented using MATLAB and tested with various images. The observations for four sample images are as discussed below.

Fig.5.12 (a) is the Airplane image with a size of 192 kilo bytes. Fig.5.12 (b)-(e) are the stego-images in five, six, seven and eight neighbor correlation methods respectively with 5120 bytes of data hidden in each and (f)-(j) are their histograms respectively.

Fig.5.13 (a) is the Lena image with a size of 535 kilo bytes. Fig.5.13 (b)-(e) are the stego-images in five, six, seven and eight neighbor correlation methods respectively with 10240 bytes of data hidden in each and (f)-(j) are their histograms respectively.

Fig.5.14 (a) is the Pepper image with a size of 768 kilo bytes. Fig.5.14 (b)-(e) are the stego-images in five, six, seven and eight neighbor correlation methods respectively with 20480 bytes of data hidden in each and (f)-(j) are their histograms respectively.

Fig.5.15 (a) is the Baboon image with a size of 525 kilo bytes. Fig.5.15 (b)-(e) are the stego-images in five, six, seven and eight neighbor correlation methods respectively with 10240 bytes of data hidden in each and (f)-(j) are their histograms respectively.
Steganography Using Pixel Value Differencing

Fig. 5.12 (a) Airplane image; (b), (c), (d), (e) stego-images; (f), (g), (h), (i), (j) histograms
Fig. 5.13 (a) Lena image; (b), (c), (d), (e) stego-images; (f), (g), (h), (i), (j) histograms
Fig. 5.14 (a) Pepper image; (b), (c), (d), (e) stego-images; (f), (g), (h), (i), (j) histograms
Fig. 5.15 (a) Baboon image; (b), (c), (d), (e) stego-images; (f), (g), (h), (i), (j) histograms
It has been observed from these figures that the distortions resulted from embedding are imperceptible to human vision. It means that such distortions are not noticeable because changes in edge areas of images are generally less conspicuous to human eye. The histograms of original images and their stego-images are completely identical, which indicates that histogram based attacks are not possible. The PSNR values of the proposed five, six, seven, and eight neighbor methods are compared with that of the proposed two, three, and four sided side match methods in Table 5.5. The lesser is the PSNR means more is the distortion. With same amount of hidden data the distortions are slightly more in two, three, and four sided methods compared to five, six, seven, and eight neighbor methods.

The performance of various steganographic methods can be rated by the three parameters; (i) security, (ii) capacity and (iii) imperceptibility. As we can see from Fig.5.8-5.11 that approximately 40% of pixels in five neighbor, 35% of pixels in six neighbor, 20% of pixels in seven neighbor and 20% of pixels in eight neighbor can be embedded; thus the hiding capacity is moderate. Stego-images should not have severe visual artifacts. In Fig.5.12-5.15 it can be observed that the stego-images are looking very innocuous. The distortion in quality of the stego-images can not be noticeable by human eye.

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Amount of hidden data (bytes)</th>
<th>PSNR values (in dB) of the proposed side match methods</th>
<th>PSNR values (in dB) of the proposed five, six, seven &amp; eight neighbor methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Two sided match</td>
<td>Three sided match</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>5120</td>
<td>47.35</td>
<td>47.34</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>10240</td>
<td>49.33</td>
<td>49.03</td>
</tr>
<tr>
<td>Pepper</td>
<td>768</td>
<td>20480</td>
<td>46.74</td>
<td>46.60</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>10240</td>
<td>41.64</td>
<td>41.61</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>46.26</td>
<td>46.14</td>
</tr>
</tbody>
</table>

In Table 5.6 the correlation values for the sample images are recorded, in all the cases it is approximately 0.9999. That means after hiding a large amount of information the stego-images, are similar to the cover images. This proves the efficacy of the proposed schemes. The estimated K-L Divergence values shown in Table 5.7 are also impressive. In all the cases this K-L divergence values are very less. The hiding capacities of the proposed schemes are shown in
Chapter 5

Table 5.8. Although these values are lesser than two, three, and four sided methods, but not very less. To improve upon the PSNR values the hiding capacity has been sacrificed.

Table 5.6 Estimated Correlation values

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Amount of hidden data (bytes)</th>
<th>Correlation values of proposed five, six, seven, and eight neighbor methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Five neighbor scheme</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>5120</td>
<td>0.9994</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>10240</td>
<td>0.9999</td>
</tr>
<tr>
<td>Pepper</td>
<td>768</td>
<td>20480</td>
<td>0.9999</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>10240</td>
<td>0.9996</td>
</tr>
</tbody>
</table>

Table 5.7 Estimated K-L Divergence values

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Amount of hidden data (bytes)</th>
<th>K-L Divergence values of proposed five, six, seven, and eight neighbor methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Five neighbor scheme</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>5120</td>
<td>35.55</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>10240</td>
<td>21.27</td>
</tr>
<tr>
<td>Pepper</td>
<td>768</td>
<td>20480</td>
<td>40.63</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>10240</td>
<td>12.73</td>
</tr>
</tbody>
</table>

Table 5.8 Estimated Capacity

<table>
<thead>
<tr>
<th>Image name</th>
<th>Image size (KB)</th>
<th>Hiding capacity of proposed five, six, seven, and eight neighbor methods (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Five neighbor scheme</td>
</tr>
<tr>
<td>Airplane</td>
<td>192</td>
<td>194537</td>
</tr>
<tr>
<td>Lena</td>
<td>535</td>
<td>458103</td>
</tr>
<tr>
<td>Pepper</td>
<td>768</td>
<td>718663</td>
</tr>
<tr>
<td>Baboon</td>
<td>525</td>
<td>765733</td>
</tr>
</tbody>
</table>
Fig. 5.16 (a)-(d) represents the variation of PSNR value with payload for Airplane image. In Fig. 5.16, the graph (a) is for five neighbor scheme, the graph (b) is for six neighbor scheme, the graph (c) is for seven neighbor scheme, and the graph (d) for eight neighbor scheme. From the graphs we can observe that if we go on increasing the payload up to a certain maximum allowable value, the PSNR may not reach less than 40. A PSNR value more than 40 represents that the distortion in stego-image is acceptable and can not be easily detected.

Fig. 5.16 Variation of PSNR with payload for Airplane image in (a) five-neighbor, (b) six-neighbor, (c) seven-neighbor, and (d) eight-neighbor methods
5.4 Comparison

In both these PVD techniques the same idea (i.e. the correlation of target pixel with neighboring pixels) is used to decide the number of bits to be embedded in a target pixel.

The comparison study of all the seven schemes of technique 1 and technique 2 reveals that, with same capacity, the distortion is in decreasing order from two neighbor scheme to eight neighbor scheme i.e. the PSNR is in an increasing order from two neighbor scheme to eight neighbor scheme.

As per the sampling arrangement of the pixels in Chang and Tseng’s [31] three and four sided side match schemes only 50% of the pixels can be targeted, but in our two, three and four sided side match schemes all the pixels except the pixels of first row, first column and last column can be targeted. Therefore, our schemes in technique 1 (two sided, three sided and four sided side match schemes) can have higher embedding capacity than that of Chang and Tseng’s side match schemes [31].

5.5 Conclusion

In technique 1, the improved versions of two sided, three sided and four sided side match schemes have been proposed, wherein the FOBP and FIEP conditions are addressed. The number of bits embedded in a target pixel is decided depending upon the correlation of the target pixel with its neighboring pixels. The embedding capacity of the proposed schemes is very good. After the information is embedded the change in quality of the images are not noticeable. The observed PSNR values are also good. The distortion is lesser in four sided side match scheme compared to the other two schemes.

In technique 2, the five, six, seven and eight neighbor correlation schemes have been proposed. FOBP and FIEP conditions are taken care at both sender and receiver side. The embedding capacities of the proposed schemes are also good. After the information is embedded the degradation in quality of the images are not noticeable. The observed PSNR values are also acceptable.