Adjacent Pixel Values Blocking and Prediction Error Expansion Based High Fidelity Reversible Data Hiding Scheme

Satyajit De
Assistant Professor
Department of Computer Science,
Maheshtala College, Budge Budge Trunk Road,
Kolkata-700141, West Bengal, India.

Abstract
This paper presents a novel reversible data hiding method based on pixel value blocking and prediction-error expansion. A cover image is divided into non-overlapping sub-blocks of two pixels. Watermark bits are embedded in two phases. In Phase-I, for each sub-block, second pixel value is predicted by the first pixel value and depending on prediction-error within a threshold limit secret bit is embedded into second pixel. Also secret bit is embedded into first pixel just by adjusting the location map without effecting into the pixel value. Then another compressed location map value is used to indicate overflow/underflow or the sub-blocks are outer threshold limit. Again in Phase-II, for each sub-block, first pixel value is predicted by the second pixel value and by depending on prediction-error within a threshold limit secret bit is embedded into first pixel. Also secret bit is embedded into second pixel by adjusting location map without effecting into the pixel value and another compressed location map value is used to indicate overflow/underflow or the sub-blocks of outer threshold limit. All secret bits can be recovered and restored the cover image completely from watermarked image. Experimental result of comparison of this scheme with recent existing scheme using different standard images shows that the embedding capacity with visual quality and PSNR values of the proposed scheme is larger than the existing scheme.

Keywords: Reversible data hiding, Prediction-error expansion, Threshold limit, Compressed location map, Peak signal-to-noise ratio (PSNR).

INTRODUCTION
Today large amount of digital data like text, images, audios and videos are transmitted using internet with the growth of information and communication technology [1,2]. Unauthorized users or attackers can easily alter, copy, delete or tamper these information during the transmission. Such problems can have recovered by using Watermark or Digital signature. Digital watermarking is a method of embedding secret data (or watermark data) into the digital multimedia content in such a way that the marked signal is perceptually indistinguishable from original cover image [3].

Reversible data hiding is a process of “lossless data hiding”. It is a special type of fragile digital data hiding scheme that can extract all hidden secret bits from watermarked media and can recover the watermarked media in its original form without loss of any information. In the spatial domain categorically RDH can be divided as DE (difference expansion), lossless compression, HS (histogram shifting) [4] and also PEE (prediction error expansion) [5].

Many reversible data hiding scheme with lossless data compressions method are proposed by Fridrich et al. [6] and Celik et al. [7] and others. In this scheme compressed watermark bits are stored by replacing some pixels of a cover image. To decrease the visual distortion compressed watermark bits are stored by replacing LSB of cover images [7].

To get a minimum image distortion with high embedding capacity DE techniques are used. Varieties difference expansion reversible data hiding schemes are proposed in [8,9,10,11]. The concept of difference expansion is first introduced by Tian [8]. Alattar [9] applied DE technique to embedded secret bits in adjacent pixel blocks. To increase the embedding capacity, Al-Qershi et al. [11] used two-dimensional difference expansion technique (2D-DE) with a threshold value depending on the image behaviour.

A lossless reversible data hiding method based on histogram modification is first proposed by Ni et al. [12]. In this method of histogram bin shifting peak point and zero point of the histogram of the image are used. The multilevel histogram technique is used by Zhao et al. to embed more secret bits [13]. In his method to enhance embedding capacity secret bits are modulated by using more peak points. Huang et al. proposed another reversible watermarking scheme with histogram bin shifting technique [14]. Luo et al. generate a strong connection among different pixel blocks to produce a difference histogram and multi-level histogram shifting to embed the secret data [15].

Thodi and Rodriguez [16] proposed a reversible data hiding method with prediction-error expansion. Hu et al. [17] introduced a modify reversible data hiding scheme by reducing the overflow location map. Li et al. proposed an improvement by using adaptive embedding and pixel selection [18]. Lee et al. proposed a reversible data hiding scheme that is free of location map and a corresponding predictive value is derived from the average of its adjacency pixels to make little bit predictive errors [19].

To get low distortion and high embedding capacity, Jaiswal et al. [20] proposed an additive prediction error based reversible data hiding technique. It is an interpolation based method that predicts pixels using varieties structured predictors in different order. Kumar et al’s represent [21] a reversible data hiding
scheme depending on prediction error expansion using adjacent pixels. According to [21], every even column pixels of a cover image are predicted by the average value of the corresponding adjacent odd column pixel values and then embedded watermark bits by expanding the corresponding prediction error. They used location map values to maintain overflow/underflow of the even column pixels and store the compressed location map into the odd column pixels after completion of watermark bits embedding in Case-I. Similarly, in Case-II every odd column pixels of a cover image are predicted by the average value of the corresponding adjacent even column pixels and watermark bits are embedded by expanding the corresponding prediction error. Also they used location map to maintain overflow/underflow of the corresponding odd column pixel and store the compressed form of it into the even column pixels after completion of watermark bits embedding. For data extraction in both cases, uncompressed the location map and skip the pixels for which location map value is 1, i.e., overflow or underflow occurred and then again by calculating the prediction errors by using the same procedure as in embedding extract the watermark bits and restore the watermarked pixels as in original form.

But in Kumar et al’s method [21], still there is a space available to improve the embedding capacity as well as the image quality for every odd pixel column in Case-I and for every even column pixel in Case-II. No threshold value is used to improve PSNR value and location map value is assigned for all pixels for which compression bits’ length becomes large.

Now a novel reversible data hiding scheme is proposed which improve Embedding capacity, image quality and PSNR value than the Kumar et al’s method [21] and Jaiswal et al’s method [1] in the following section.

In the proposed method based on pixel-value blocking and prediction error expansion I establish a better reversible watermarking scheme than existing scheme. In our method the cover image is divided into two pixels non-overlapped sub-blocks and embeds the secret bits in two phases.

In Phase-I, I embed the watermark bits into first pixel of every sub-block just by adjusting the location map value (either 0 or 1) and I keep unchanged the pixel to increase the image quality. Next embed watermark bits into the second pixels (original pixel) of every sub-block by using prediction error expansion technique considering first pixels as predicted pixels. I skip those sub-blocks from embedding in which either prediction error exceeds the threshold value or overflow/underflow occurred and set location map value as 2. In this Phase compressed location map stores using the first pixels of every sub-block.

Similarly, in Phase-II, I embed watermark bits into the first pixels (original pixel) of every sub-block by using prediction error expansion technique considering second pixels as predicted pixels. Skip the sub-blocks from embedding in which either prediction error exceeds the threshold value or overflow/underflow occurred and set the location map value to 2 to indicate it. Again embed the watermark bits into second pixels of every sub-block just by adjusting the location map value (either 0 or 1) and keep unchanged that pixel to increase the image quality. In this Phase the compressed location map stores using the second pixels of every sub-block.

Decompress the location map in each Phase and skip the blocks for which location map value is 2. Then in data extraction and image restoration procedure first I have to extract those watermark bits which are embedded at Phase-II and restore the pixels as in watermarked pixel of Phase-I and then extract the watermark bits which are embedded at Phase-I and restore the watermarked pixels of Phase-I pixels as in original form. As a result, embedding capacity and image quality becomes high in this proposed scheme.

RELATED WORK

Here I briefly describe about prediction error expansion based reversible data hiding scheme which hides one secret bit into the pixel of cover image at a time. Suppose the watermark bits $b_j = 0$ or 1 will have to be embed at the pixel $s_{jk}$ at $(j,k)^{th}$ position. For that after finding the corresponding predicted pixel say $s_{jk}'$, calculate the prediction error as $e_{jk} = s_{jk} - s_{jk}'$. Now the watermark bits $b_j$ are embedded by expanding the prediction error $e_{jk}$ as $e_{jk}' = 2 \times e_{jk} + b_j$. Then the watermarked pixels are calculated as $s_{jk}'' = s_{jk} + e_{jk}'$. The overflow or underflow situations are handled by using location map.

In extraction procedure watermarked bits are extracted as well as original image is also recovered. Suppose the watermarked pixel $s_{jk}''$ is predicted as $s_{jk}'''$. Now the prediction error is calculated as $e_{jk}''' = s_{jk}'' - s_{jk}'''$. Watermark bits are extracted as $b_j = \lfloor e_{jk}''' / 2 \rfloor$. Calculate the original prediction error as $e_{jk} = e_{jk}''' + e_{jk}'$. Kumar et al’s utilized this concept in [21] for embedding and extracting watermark bits.

Kumar et al’s method [21]

Kumar et al’s represent [21] a reversible data hiding scheme depending on prediction error expansion using adjacent pixels.

Case-I

Embedding procedure

According to the Kumar et al’s method [21], every even column pixels of a cover image are predicted by the average value of the corresponding adjacent odd column pixel values. Prediction error is calculated for every even column pixel as $e = \text{original pixel} - \text{predicted value}$. Embed watermark bits by expanding prediction errors as $e'' = 2 \times e + b$, where $b$ is the watermark bit either 0 or 1 and $e''$ is the modified prediction error. Watermark pixel value is calculated as $s = \text{predicted value} + \text{modified prediction error}$. If $s < 0$ or $s > 255$ then set the location map $\text{lm}=1$ else $\text{lm}=0$. Using arithmetic
coding compressed the location map and embed into odd columns and ultimately watermarked image is generated.

**Extracting Procedure**

Uncompressed the location map and skip the pixels for which location map value is 1, i.e. overflow or underflow occurred. Then applying the following procedure extract the watermark bits and restore the original pixels. Every even column pixels of the watermarked image are predicted by the average value of the corresponding adjacent odd column pixels values. Prediction error is calculated for every even column pixel as \( e = \text{watermarked pixel value} - \text{predicted value} \). Extracted watermark bit \( b \) as \( b = e \mod 2 \). Calculate the original prediction error as \( e' = \frac{e}{2} \). Restore the original pixel value as original pixel value = predicted value + \( e' \).

**Case-II**

Similarly, every odd column pixels of a cover image are predicted by the average value of the corresponding adjacent even column pixel values to embed watermark bits in second case. As using the same procedure as previous extract the watermark bits and restores the original pixels.

But in Kumar et al's method [21], still there is a space available to improve the embedding capacity as well as the image quality for every odd column pixel in Case-I and for every even column pixel in Case-II. No threshold value is used to improve PSNR value and location map value is assigned for all pixels for which compression bits’ length becomes large.

Now a novel reversible data hiding scheme is proposed which improve Embedding capacity, image quality and PSNR value with less compressed location map bits’ length than the Kumar et al’s method [21] and Jaiswal et al’s method [20] in the following section.

**PROPOSED SCHEME**

In this proposed scheme embedding and extracting procedures are as follows:

First the cover image is divided into two pixels non-overlapped sub-blocks and embeds the secret bits in two phases.

In Phase-I embedding scheme, I embed the secret bits into first pixel of every sub-block just by adjusting location map value. For embedding the bit ‘0’ or ‘1’ into first pixel of every sub-block I set the location map value to 0 or 1 respectively. Here first pixel is keep unchanged to get better image quality i.e. high PSNR value. Now embed secret bits into the second pixel (original pixel) of every sub-block by considering the first pixel value as the predicted pixel. Prediction error is calculated by using second pixel value and predicted pixel value. To decreases the distortion of current pixels I compare the prediction error with threshold value. If the absolute value of prediction error is exceeded the threshold limit (say T1) then set the location map value to 2 and skip the sub-block from embedding otherwise prediction error is expanded using watermark bits which I have to embed. Watermark pixel is calculated for second pixel by using it and expanded prediction error. If overflow or underflow occurred in watermarked pixel, then set the location map value to 2 and restore the original pixel and skip the sub-block from embedding. After embedding all watermark bits in Phase-I compress the location map by arithmetic coding and stored the compressed bits using first pixels of every sub-block.

Figure-3 shows the embedding procedure for Phase-I in different sub-blocks.

Again in Phase-II, I embed the secret bits into second pixel of every sub-block just by adjusting location map value as Phase-I.

Also in Phase-II embedding scheme, secret bits are embedded into the first pixel (original pixel) of every sub-block by considering the second pixel value as the predicted pixel applying the same procedure as Phase-I by considering threshold value \( T_2 = 10 \). After embedding all watermark bits in Phase-II compress the location map by arithmetic coding and stored the compressed bits using second pixels of every sub-block. Figure-4 shows the embedding procedure for Phase-II in different sub-blocks.

If I increase the threshold value \( T_1 \) or \( T_2 \) then embedding capacity becomes increases but image quality becomes decreases, so I have to choose proper threshold values to ensure good image quality and high embedding capacity. Experimental results shown in Table-1, Table-2 and Table-3 of the proposed scheme for different standard images considered the threshold values \( T_1 = 10 \) (in Phase-I) and \( T_2 = 10 \) (in Phase-II).

For data extraction and image restoration procedure first I have to extract those watermark bits which are embedded at Phase-II and restore the pixels as in watermarked pixel of Phase-I and then extract the watermark bits which are embedded at Phase-I and restore the watermarked pixels of Phase-I pixels as in original form. First watermarked image is divided into two pixels non-overlapped sub-blocks and extract the secret bits in two phases.

Decompress the compressed location map of Phase-II and skip the sub-blocks from extraction whose location map value is 2. Now extract the watermark bits using following procedure.

Prediction error is calculated for every sub-block by considering first pixel as original pixel and second pixel as predicted pixel. Extracted bit from first pixel is \( b = \text{Prediction error mod 2} \). Original prediction error is calculated as \( e = \frac{\text{Prediction Error}}{2} \). Now watermarked pixel of Phase-I is restored by adding predicted value with e. Now extract the secret bits from second pixel of every sub-block using location map values.

Similarly decompress the compressed location map of Phase-I and skip the sub-blocks from extraction whose location map value is 2. Now extract the secret bits from first pixel of every
sub-block using location map values. Again to fetch the secret bits from second pixel Prediction error is calculated for every sub-block by considering second pixel as original pixel and first pixel as predicted pixel. Extracted bit from second pixel is \( w = \text{Prediction error} \mod 2 \). Original prediction error is calculated as \( e_1 = \frac{\text{Prediction Error}}{2} \). Now original pixel is restored by adding predicted value with \( e_1 \).

Secret bits’ extraction and restoring the original pixels of Phase-II and Phase-I are shown in Fig: 5 and Fig: 6 respectively. The embedding and extracting algorithms are as follows:

## Data Embedding Procedure

### Algorithm for data embedding

**Input:** i) A cover image of order \( m \times n \)

ii) Secret bit sequence for embedding into the cover image.

**Output:** A watermarked image of order \( m \times n \)

**Step 1:** A cover image ‘Pic’ of order \( m \times n \) is taken from user and divided this cover image into two pixels non-overlapping sub-blocks. Now I embed the secret bits in two phases.

**Phase-I**

**Step 2:** In every sub-block I embed two watermark bits. Here I consider the location map array as ‘lmap’

**Step 3:**

a) Consider a generalized \( i^{th} \) sub-block as \((s_i, s_{i+1})\).

b) To decrease the distortion and increase the embedding capacity of cover image, a watermark bit says \( b_i=0 \) or 1 is embedded in the first pixel \( s_i \) just by adjusting the location map value and keeping unchanged that pixel value.

If \( b_i=0 \) then consider ‘0’ is embedded at \( s_i \) and set location map value \( lmap(i)=0 \) otherwise consider ‘1’ is embedded and set the location map value \( lmap(i)=1 \). Next I embed another bit into second pixel \( s_{i+1} \) using following procedure.

c) Here the prediction error is calculated for second pixels\( s_{i+1} \) by considering \( s_i \) as the predicted pixel value.

d) Prediction error \( e_i \) is calculated as \( e_i = \text{original pixel value} – \text{predicted pixel value}=s_{i+1} – s_i \)

e) If absolute value of prediction error exceeds the threshold value then more distortions occurred and reject this sub-block keeping unchanged of pixel values and set the location map value to 2 as follows:

If \( |e_i| > T1 \) then \( lmap(i)=2 \) and continue from Step 4 for next sub-block. Here \( T1 \) is the threshold value for Phase-I.

f) Consider another watermark bit say \( b_{2i} \) embedded into second pixel \( s_{i+1} \) by expanding the prediction error \( e_i \) such as \( e_i = e_i + b_{2i} \). Here \( e_i \) is the modified prediction error.

\[ g) \text{Calculate a value ‘}v\text{’ by adding modified prediction error with original pixel value as } v = s_{i+1} + e_i. \]

\[ h) \text{If } v > 255 \text{ or } v < 0 \text{ then overflow or underflow occurred and set the location map value as } lmap(i)=2 \text{ and reject this } i^{th} \text{ sub-block from data embedding and continue from Step 4.} \]

i) Now I get the watermarked pixel value \( ass_{i+1} = v \) and the watermarked sub-block is \((s_i’, s_{i+1}’)\) for Phase-I. Here \( s_i’ = s_i \).

**Step 4:** Repeat the same procedure for next sub-block by setting \( i=i+1 \) and continuing from step 3 until all data bits are embedded.

**Step 5:** Now the array of location map values “lmap(\( i \))” is lossless compressed by arithmetic coding and store the compressed location map bits with auxiliary information (length of the compressed location map bits) into first pixels of every sub-blocks and I get the watermarked image for Phase-I.

In this watermarked image again I embed more secret bits using the algorithm of Phase-II and I get final watermarked image as follows.

**Phase-II**

**Step 6:** After embedding the watermark bits in Phase-I again I embed watermark bits in Phase-II in every sub-block. Firstly, watermark bit will be embedding at the first pixel by using the second pixel as predicted pixel and then another watermark bit will be embedding at the second pixel directly by setting the location map value. Here I consider the location map array as ‘lm’

**Step 7:**

a) Consider a generalized \( i^{th} \) sub-block as \((s_i’, s_{i+1}’)\). Here the first pixel \( s_i \) is the original pixel value and the predicted pixel value is \( s_{i+1} \).

b) Prediction error \( e_i \) is calculated as \( e_i = s_i’ – s_{i+1}’ \)

c) If absolute value of prediction error exceeds the threshold value then more distortions occurred and reject this sub-block keeping unchanged of pixel values and set the location map value to 2 as follows:

If \( |e_i| > T2 \) then \( lm(i)=2 \) and continue from Step 8 for next sub-block. Here \( T2 \) is the threshold value for Phase-II.

\[ d) \text{Consider a watermark bit say } p_i \text{ has to be} \]

\[ \text{embedded into first pixel } s_i’ \text{ by expanding the prediction error } e_i \text{ such as } e_i = e_i + p_i. \text{ Here } e_i \text{ is the modified prediction error.} \]

\[ e) \text{Calculate a value ‘}m\text{’ by adding modified prediction error with original pixel value as } m = s_i’ + e_i. \]

\[ g) \text{If } m > 255 \text{ or } m < 0 \text{ then overflow or underflow occurs and set the location map value as } lm(i)=2 \text{ and reject this } i^{th} \text{ sub-block from data embedding and continue from Step 8.} \]

\[ f) \text{Final watermarked pixel value is } s_i’’ = m \]
h) To decrease the distortion and increase the embedding capacity of cover image, another watermark bit say $p_2$ is embedded in the second pixel $s_{i+1}$ just by adjusting the values of location map without any effecting on that pixel as follows.

If $p_2 = 0$ then consider '0' is embedded at $s_{i+1}$ and set location map value $lm(j)=0$ otherwise consider ‘1’ is embedded and set the location map value $lm(j)=1$.

i) Now I get the watermarked sub-block as $(s_i^\prime, s_{i+1}^\prime)$ from Phase-II. Here $s_{i+1}^\prime = s_{i+1}$.

Step 8: Repeat the same procedure for next sub-block by setting $i=i+1$ and continuing from step 7 until another all data bits are embedded.

Step 9: Now the array of location map values “$lm()$” is lossless compressed by arithmetic coding and store the compressed location map bits with auxiliary information (length of the compressed location map bits) into second pixels of sub-blocks and I get the final watermarked image say “Pic’”.

**NUMERICAL EXAMPLES FOR DATA EMBEDDING**

For example, consider a sample data pixels of order 2×4 as a cover image shown in the following fig. 1. Divide this cover image into two pixels non-overlapping sub-blocks are shown in fig. 2. In every sub-block four watermark bits is embedded (two bits in Phase-I and two bits in phase-II) or ignore the block if overflow or underflow occurred in the corresponding phase.

```
| 205 | 195 |
| 240 | 250 |
```

Fig 1: Cover image of order 2×4

```
| 205 | 195 |
| 147 | 150 |
| 240 | 250 |
| 185 | 185 |
```

Fig 2: Divide the cover image into 2 pixels’ sub-blocks

In Phase-I considering threshold value $T_1=10$ (shown in fig. 3), I embed the secret bits in different sub-block. For example, in Fig. 3(a), a watermark bit says ‘1’ is inserted into the first pixel of the sub-block (Sub-block-1) by setting the location map value $lm(i)=1$, here $i$ is the sub-block number. To embed the watermark bit into second pixel of Sub-block:1, I calculate the prediction error $195 - 205 = -10$ (considering 205 as predicted value). Science $|-10| \leq$ threshold value $T_1(=10)$ therefore expand the prediction error by adding watermark bit say ‘1’ into it. Expanded the prediction error $= -10 + 1 = -9$. Now watermark pixel for Phase-I = second pixel + expanded prediction error $= 195 + (-9) = 186$. In fig. 3(c) watermark pixel value is greater than 255 therefore overflow occurred and I skip this sub-block (Sub-block: 3) from data embedding by setting the location map value $lm(i)=2$. Fig. 3(b) shows data embedding using the same procedure in Sub-block: 2. Also in Sub-block: 4 data will be embedded as previous procedure.

```
| 205 | 195 | 147 | 150 |
```

Fig: 3 (a)

```
| 147 | 150 |
| 240 | 250 |
| 185 | 185 |
```

Fig: 3 (b)

```
| 205 | 195 |
| 147 | 150 |
| 240 | 250 |
| 185 | 185 |
```

Fig: 3 (c)

Fig 3: Watermark bits embedding in Phase-I taking threshold value $T_1=10$

In Phase-II (shown in figure 4) for every sub-block, I embed the secret bits by using threshold value $T_2=10$. For example,
in figure 4(b) (for Sub-block: 2), to embed a watermark bit into first pixel, I calculate the prediction error $147 - 154 = -7$ (considering 154 as predicted value). Science $[-7] \leq$ threshold value $T_2 = 10$ therefore expand the prediction error by adding watermark bit say ‘0’ into it. Expanded prediction error $= -7 + 0 = -7$. Now watermark pixel for Phase-II = first pixel + expanded prediction error $= 147 + (-7) = 140$.

Another watermark bit says ‘1’ is inserted into the second pixel of the sub-block by setting the location map value $l_{m}(i)=1$, here $i$ is the sub-block number. In Figure 4(a) (for Sub-block: 1) since absolute value of prediction error exceeds the threshold value $T_2=10$ therefore skip this sub-block from data embedding for Phase-II and set the location map value $l_{m}(i)=2$, here $i$ is the block number. Fig. 4(c) shows an example to embed secret bits in Sub-block: 3. Similarly, secret bits can be inserted into Sub-block: 4.

Data Extraction Procedure

Algorithm for data extraction and image recovery

**Input:** A watermark image Pic’ of order $m \times n$  

**Output:** First extract the watermark bits which are embedded in Phase-II and then from Phase-I and restore the original cover image of the same order.

**Step 1:** Divided this watermarked image into two pixels non-overlapping sub-blocks.

**Watermark bits’ extraction and image recovery of Phase-II embedding procedure**

**Step 2:** Decompress the compressed location map “lm” whose bits are stored using the second pixels of the sub-blocks in Phase-II. Now for every sub-block do the following to extract watermark bits embedded in Phase-II and restore the pixels in the form as in Phase-I.

**Step 3:**

a) Consider a generalized watermarked $i^{th}$ sub-block as $(s_{i\&}, s_{i+1\&})$. Here the first pixel $s_{i\&}$ is the original pixel value and the predicted pixel value is $s_{i+1\&}$.

b) If $l_{m}(i)=2$ then this sub-block does not contain any secret bits, therefore skip this sub-block and continue from Step 4.

c) Prediction error is calculated for every sub-block by considering first pixel as original pixel and second pixel as predicted pixel. Here Prediction error $er_i = |s_{i\&} - s_{i+1\&}|$. Extracted bit from first pixel is $p_1 = [\text{Prediction error mod } 2] = |er_i mod 2|$. 

d) Original prediction error is calculated as $er_i = \left[\frac{\left|\text{Prediction Error} - \text{Watermarked pixel} \times \text{Original pixel} + \text{Modfied Prediction Error}\right|}{2}\right]$. Now watermarked pixel of Phase-I is restored by adding predicted value with original prediction error $er_i$, i.e. the watermarked pixel as in phase-I is $s_{i\&} = s_{i+1\&} + er_i$.

e) Now extract the watermark bit from second pixel $s_{i+1\&}$ of this sub-block using location map value. If $l_{m}(i)=1$ then the extracted bit is $p_2=1$ else if $l_{m}(i)=0$ then the extracted bit is $p_2=0$ and the watermarked pixel as in Phase-I is $s_{i+1\&} = s_{i+1\&}$.

**Step 4:** Repeat the same procedure for next sub-block by setting $i = i + 1$ and continuing from step 3 until all watermark bits are extracted which are embedded in Phase-II.

**Watermark bits’ extraction and image recovery of Phase-I embedding procedure**

**Step 5:** Decompress the compressed location map “lmap” whose bits are stored using the first pixels of the sub-blocks in Phase-I. Now for every sub-block do the following to extract watermark bits embedded in Phase-I and restore the original pixels.

**Step 6:**

a) Consider a generalized watermarked $i^{th}$ sub-block as $(s_{i\&}, s_{i+1\&})$.

b) If $l_{m}(i)=2$ then this sub-block does not contain any secret bits, therefore skip this sub-block and continue from Step 7.

c) Extract the watermark bit from first pixel $s_{i\&}$ of this sub-block using location map value. If $l_{m}(i)=1$ then the
extracted bit is $b_1=1$ otherwise if $lmap(i)=0$ then the extracted bit is $b_1=0$ and restore the original pixel as $s_i=s_i^\prime$.

d) Now extract the watermark bit from second pixel $s_{i+1}^\prime$. Here the second pixel $s_{i+1}^\prime$ is the original pixel value and the predicted pixel value is $s_i^\prime$. Prediction error is calculated for this sub-block by considering second pixel as original pixel and first pixel as predicted pixel. Calculate the Prediction error $e_i = s_{i+1}^\prime - s_i^\prime$. Extracted bit from second pixel is $b_{2}\equiv |\text{Prediction error mod } 2| = |e_i\text{ mod } 2|$. 

e) Original prediction error is calculated as $e_i^\prime = \left[ \frac{\text{Prediction Error}}{2} \right] = \left[ e_i \right]$. Now original pixel is restored by adding predicted value with the original prediction error $e_i^\prime$, i.e. the original pixel is $s_{i+1} = s_i^\prime + e_i^\prime$.

**Step 7:** Repeat the same procedure for next sub-block by setting $i = i + 1$ and continuing from step 6 until all watermark bits are extracted which are embedded in Phase-I.

**NUMERICAL EXAMPLES OF DATA EXTRACTION**

In extraction procedure (shown in figure 5) I extract all watermark bits which are embedded at Phase-II embedding procedure. Decompress the compressed location map “lm” of Phase-II and skip the sub-blocks from extraction whose location map value is 2.

In figure 5(a) since the location map value $lm(i)=2$ therefore this sub-block (Sub-block: 1) not contains any watermark bits and skip it from extracting.

In figure 5(b) since the location map value $lm(i)=1$ therefore this sub-block (Sub-block: 2) contains two watermark bits. Now extract the watermark bits using following procedure. Prediction error is calculated for this sub-block by considering first pixel $s_i^\prime=140$ as watermarked pixels and second pixel $s_{i+1}^\prime=154$ as predicted pixel i.e. prediction error $e_i^\prime = \text{watermarked pixel} – \text{predicted pixel} = s_i^\prime - s_{i+1}^\prime = 140 - 154 = -14$. Extracted bit from first pixel is $b_1 = |\text{Prediction error mod } 2| = | -14 \text{ mod } 2 | = 0$. Original prediction error is calculated as $e_i^\prime = \left[ \frac{\text{Prediction Error}}{2} \right] = \left[ -14 \right] = -7$.

Now watermarked pixel of Phase-I is restored by adding predicted value with $e$ i.e. Recovered pixel as in Phase-I $s_i^\prime = \text{predicted pixel} + \text{original prediction error}= s_i^\prime + e_i^\prime = 140 + (-7) = 147$. Now extract the secret bits from second pixel of this sub-block using location map value. Since $lm(i)=1$ therefore watermark bit for second pixel is $b_{2}=1$. Since second pixel is kept unchanged therefore recovered pixel is $s_{i+1} = 154$. Figure 5(c) also shows the watermark bits’ extraction and image restoration up to the pixels which I get after Phase-I embedding for Sub-block: 3.

In extraction procedure (shown in figure 6) I extract all watermark bits which are embedded at Phase-I embedding procedure. Decompress the compressed location map “lm” of Phase-I and skip the sub-blocks from extraction whose location map value is 2.

In figure 6(c) since the location map value $lm(i)=2$ therefore this sub-block (Sub-block: 3) not contains any watermark bits and skip it from extracting.

In figure 6(a) since the location map value $lm(i)=1$ therefore this sub-block (Sub-block: 1) contains two watermark bits. Now extract the first watermark bit from first pixel $s_i^\prime$ as since $lm(i)=1$ therefore the extracted watermark bit is $b_{2}=1$. Since first pixel is kept unchanged therefor recovered original pixel is $s_i = 205$. Now second watermark bit is extracted from second pixel as follows:

**Fig 5:** Extract watermark bits embedded in Phase-II and recover the pixels in the form which was exists as in Phase-I.
Fig. 6(a) Since location map value Imap(1)=1 therefore extracted watermark bit \( p_1 \) = 1. Since this pixel is keeping unchanged therefore the recovered pixel value is same with this pixel value.

Water marked sub-block of phase-I with location map value Imap(1)=1

\[ \begin{align*}
& s_1 \quad s_2 \\
& 205 \quad 186
\end{align*} \]

Extract watermark bit from first pixel using location map

Extracted bit sequence is 1

Original pixel sub-block

\[ \begin{align*}
& s_1 \quad s_2 \\
& 205 \quad 195
\end{align*} \]

Extracted watermark bit from second pixel and recover the original pixel value


Fig. 6(b) Since location map value Imap(1)=0 therefore extracted watermark bit \( p_1 \) = 0. Since this pixel is keeping unchanged therefore the recovered pixel value is same with this pixel value.

Water marked sub-block of phase-I with location map value Imap(1)=0

\[ \begin{align*}
& s_1 \quad s_2 \\
& 147 \quad 154
\end{align*} \]

Extract watermark bit from first pixel using location map

Extracted bit sequence is 0

Original pixel sub-block

\[ \begin{align*}
& s_1 \quad s_2 \\
& 147 \quad 150
\end{align*} \]

Extract watermark bit from second pixel and recover the original pixel value


Fig. 6(c) Since Imap(1)=2 therefore no watermark bit exists in this sub-block and we skip this sub-block.

Water marked sub-block of phase-I with location map value Imap(1)=2

\[ \begin{align*}
& s_1 \quad s_2 \\
& 240 \quad 250
\end{align*} \]

Original pixel sub-block

\[ \begin{align*}
& s_1 \quad s_2 \\
& 240 \quad 250
\end{align*} \]

Fig 6: Extract watermark bits embedded in Phase-I and recover the original pixels.

Prediction error is calculated for this sub-block by considering second pixels \( s_{i+1} = 186 \) as the watermarked pixel and first pixel \( s_i = 205 \) as the predicted pixel. Here the prediction error \( e_{i} = \) watermarked pixel – predicted pixel= \( s_{i+1} - s_i = 186 - 205 = -19 \). Extracted bit from second pixel is \( p_2 = | \text{Prediction error mod 2} | = |-19 \text{ mod 2} | = 1 \). Original prediction error is calculated as \( e_{i} = \frac{| \text{Prediction Error} |}{2} = \frac{| -19 |}{2} = -10 \).

Now original pixel is restored by adding predicted value with \( e_{i} \) i.e. Recovered original pixel \( s_i = \text{predicted pixel} + e_{i} = 205 + (-10) = 195 \). Figure 6(b) also shows the watermark bits’ extraction and original image restoration for Sub-block: 2.

EXPERIMENTAL RESULTS AND COMPARISONS

The standard 512×512 grey scale cover images are taken from USC-SIPT image database, researchget.net and from google in our experiment. Experiment is implemented by using MATLAB. Watermark bits are randomly generated in MATLAB using the function randi ([0,1]). Also another (m×n) number of watermark bits can be embedded into cover image in Phase-II. So for a 512 × 512 grey scale image, maximum secret bits could be embedded in every Phase in the proposed scheme is \((512 \times 512) \times \frac{255^2}{2} = 262144 \) bits. So, maximum 262144 bits + 262144 bits = 524288 bits can be embedded into the cover image.

In the proposed scheme, since I try to embed watermark bits in every pixel of the cover image in both the phases therefore embedding capacity becomes more high than the recent existing schemes [1,2]. Also the PSNR of the marked image versus its cover image is at least \(10 \log_{10} \frac{255^2}{2} \approx 48.13 \text{dB} \).

In the proposed scheme since I skipped all the 2-pixels sub-blocks whose prediction error is more than threshold value or overflow/underflow occurred (i.e. I avoid unnecessary expansions of pixels) therefore the PSNR value becomes higher than Jaiswal et al’s method [20] and Kumar et al’s method [21]. Also PSNR becomes high due to keep unchanged of first pixels in Phase-I and keep unchanged of second pixel in Phase-II.

PICTURES BEFORE EMBEDDING (512×512 GREY SCALE IMAGES USED FOR EXPERIMENT)

- Airplane
- Barbara
- Boat
- Goldhill
- Lena
- Pepper
- Sailboat
- Tiffany
- Zelda

Fig 7: Pictures used in experiment before watermark bits embedding (All are gray scale images of size 512×512 taken from USC-SIPI image database and researchget.net and from google.)
Pictures after embedding

- Airplane (EC: 421510 PSNR: 37.05)
- Barbara EC: 365692, PSNR: 37.57
- Boat EC: 370650, PSNR: 36.15
- Goldhill (EC: 344912 PSNR: 35.68)
- Lena (EC: 423905 PSNR: 35.84)
- Pepper (EC: 383396 PSNR: 35.23)
- Sailboat (EC: 367352 PSNR: 36.79)
- Tiffany (EC: 401052 PSNR: 36.33)
- Zelda (EC: 453018 PSNR: 35.60)

Fig 8: Watermarked pictures considering threshold value T1=10 (in Phase-I) and T2=10 (in phase-II). Experimental images with watermark bits hided in them and the corresponding PSNR value.

Comparison Table of proposed scheme with existing scheme

The results of embedding capacity of proposed scheme compared with Jaiswal et al’s method [20] and Kumar et al’s method [21] using nine standards 512×512 sized grey scale images are shown in Table 1 as follows.

The above table Table-1 shows that, in Phase-I, the average embedding capacity of Jaiswal et al’s method is 208649.11 bits and Kumar et al’s method is 130560 bits, but in the proposed method it is 219598.67. This shows that proposed method hides 10949.56 bits more on average than Jaiswal et al’s method and 89038.67 bits more than Kumar et al’s method.

In Phase-II, from the above table Table-1 I see that, the average embedding capacity of Jaiswal et al’s method is 225072.44 bits and Kumar et al’s method is 261120 bits, but in the proposed method it is 392387.55. This shows that proposed method hides 167315.11 bits more on average than Jaiswal et al’s method and 131267.55 bits more than Kumar et al’s method.

On the other hand, the following table Table-2 shows that in Phase-I, average PSNR of Jaiswal et al’s method is 35.29dB and that of Kumar et al’s method is 35.37dB, but in the proposed method it is 39.59 dB. So, the proposed method improves the average PSNR 4.3 dB more than Jaiswal et al’s method and 4.22 dB more than Kumar et al’s method.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane</td>
<td>Phase-I</td>
<td>239325</td>
<td>130560</td>
<td>226800</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>243378</td>
<td>261120</td>
<td>421510</td>
</tr>
<tr>
<td>Barbara</td>
<td>Phase-I</td>
<td>211843</td>
<td>130560</td>
<td>200098</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>225116</td>
<td>261120</td>
<td>365692</td>
</tr>
<tr>
<td>Boat</td>
<td>Phase-I</td>
<td>207974</td>
<td>130560</td>
<td>210854</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>238631</td>
<td>261120</td>
<td>370650</td>
</tr>
<tr>
<td>Goldhill</td>
<td>Phase-I</td>
<td>178190</td>
<td>130560</td>
<td>204084</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>195274</td>
<td>261120</td>
<td>344912</td>
</tr>
<tr>
<td>Lena</td>
<td>Phase-I</td>
<td>223545</td>
<td>130560</td>
<td>234616</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>236375</td>
<td>261120</td>
<td>423906</td>
</tr>
<tr>
<td>Pepper</td>
<td>Phase-I</td>
<td>201132</td>
<td>130560</td>
<td>225476</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>224856</td>
<td>261120</td>
<td>383396</td>
</tr>
<tr>
<td>Sailboat</td>
<td>Phase-I</td>
<td>200319</td>
<td>130560</td>
<td>205250</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>216638</td>
<td>261120</td>
<td>367352</td>
</tr>
<tr>
<td>Tiffany</td>
<td>Phase-I</td>
<td>199194</td>
<td>130560</td>
<td>221338</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>211544</td>
<td>261120</td>
<td>401052</td>
</tr>
<tr>
<td>Zelda</td>
<td>Phase-I</td>
<td>216320</td>
<td>130560</td>
<td>247872</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>233840</td>
<td>261120</td>
<td>453018</td>
</tr>
<tr>
<td>Average value</td>
<td>Phase-I</td>
<td>208649.11</td>
<td>130560</td>
<td>219598.67</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>225072.44</td>
<td>261120</td>
<td>392387.55</td>
</tr>
</tbody>
</table>

EC: Embedded capacity

Following table Table-2 also shows that in Phase-II, average PSNR of Jaiswal et al’s method is 33.29 dB and that of Kumar et al’s method is 33.50 dB, but in the proposed method it is
36.3 dB. So, the proposed method improves the average PSNR 3.01 dB more than Jaiswal et al.’s method and 2.8 dB more than Kumar et al.’s method.

Table 2: Comparison of PSNR values of proposed scheme with existing scheme [20,21] in Phase-I and Phase-II

<table>
<thead>
<tr>
<th>Cover Images</th>
<th>Phases</th>
<th>PSNR</th>
<th>PSNR</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane</td>
<td>Phase-I</td>
<td>33.91</td>
<td>34.56</td>
<td>40.76</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>32.81</td>
<td>33.07</td>
<td>37.05</td>
</tr>
<tr>
<td>Barbara</td>
<td>Phase-I</td>
<td>33.49</td>
<td>33.89</td>
<td>40.74</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>31.50</td>
<td>31.97</td>
<td>37.57</td>
</tr>
<tr>
<td>Boat</td>
<td>Phase-I</td>
<td>36.36</td>
<td>36.10</td>
<td>39.22</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>31.70</td>
<td>32.02</td>
<td>36.15</td>
</tr>
<tr>
<td>Goldhill</td>
<td>Phase-I</td>
<td>35.86</td>
<td>35.56</td>
<td>38.48</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>34.60</td>
<td>34.89</td>
<td>35.68</td>
</tr>
<tr>
<td>Lena</td>
<td>Phase-I</td>
<td>36.58</td>
<td>36.25</td>
<td>39.52</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>33.85</td>
<td>33.45</td>
<td>35.84</td>
</tr>
<tr>
<td>Pepper</td>
<td>Phase-I</td>
<td>36.45</td>
<td>36.86</td>
<td>38.20</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>34.47</td>
<td>34.89</td>
<td>35.23</td>
</tr>
<tr>
<td>Sailboat</td>
<td>Phase-I</td>
<td>32.19</td>
<td>32.69</td>
<td>39.90</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>30.46</td>
<td>30.85</td>
<td>36.79</td>
</tr>
<tr>
<td>Tiffany</td>
<td>Phase-I</td>
<td>34.86</td>
<td>35.01</td>
<td>39.86</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>33.49</td>
<td>33.90</td>
<td>36.33</td>
</tr>
<tr>
<td>Zelda</td>
<td>Phase-I</td>
<td>37.91</td>
<td>37.39</td>
<td>39.63</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>36.73</td>
<td>36.44</td>
<td>35.60</td>
</tr>
<tr>
<td>Average value</td>
<td>Phase-I</td>
<td>35.29</td>
<td>35.37</td>
<td>39.59</td>
</tr>
<tr>
<td></td>
<td>Phase-II</td>
<td>33.29</td>
<td>33.50</td>
<td>36.3</td>
</tr>
</tbody>
</table>

PSNR: Peak signal to noise ratio.

Comparison Graphs

The Embedding capacity comparisons Graph is shown below in Fig 9.

Fig 9: Graph to show a comparison of embedding capacity of proposed scheme with existing schemes (Phase-II result).

The PSNR comparisons Graph is shown below in Fig 10.

Fig 10: Graph to show a comparison of PSNR results of proposed scheme with existing schemes (Phase-II result).

CONCLUSION

So, in this paper I proposed a new Pixel-value blocking and Prediction error expansion based reversible watermarking scheme to improve Jaiswal et al’s method [20] and Kumar et al’s method [21]. In this scheme I embedded watermark bits in two Phases. In first Phase water mark bit stored into first pixel of every sub-block by setting location map value. To embed
another watermark bit into second pixel of that sub-block by calculating the prediction error. Skip the sub-blocks if the value of prediction error exceeds threshold value otherwise store watermark bit into it. Similarly, in second Phase a watermark bit is embedded into second pixel by calculating the prediction error. Skip the sub-blocks if the value of prediction error exceeds threshold value otherwise store watermark bit into it. I embed another water mark bit into second pixel by setting location map value. In this way I get a watermarked image and finally from the watermarked image I can fetched the secret bits and restore the cover image completely. Experimental results of the proposed scheme show that the average embedding capacity and average PSNR is better than Jaiswal et al’s method [20] and Kumar et al’s method [21]. In our future work I try to embed more secret bits to improve embedding capacity with improved PSNR using Pixel value blocking and PEE technique.

REFERENCES


