International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 11 (2019) pp. 2585-2595
© Research India Publications. https://dx.doi.org/10.37622/IJAER/14.11.2019.2585-2595

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 Trunck 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 hided 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 whith 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 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 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 $er_{jk} = s_{jk} - s_{jk}'$. Now the watermark bits b_j are embedded by expanding the prediction error er_{jk} as $er_{jk}' = 2 \times e_{jk} + b_j$. Then the watermarked pixels are calculated as $s_{jk}'' = s_{jk}' + er_{jk}'$. 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 $er_{jk}'' = s_{jk}'' - s_{jk}'''$. Watermark bits are extracted as $b_j = |\, er_{jk}'' \bmod 2 \,|$. Calculate the original prediction error as $er_{jk} = \left\lfloor \frac{er_{jk}}{2} \right\rfloor$ and recover the original pixel value as $s_{jk} = s_{jk}''' + er_{jk}$. 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 err = original pixel – predicted value. Embed watermark bits by expanding prediction errors as $err' = 2 \times err + b$, where b is the watermark bit either 0 or 1 and err' is the modified prediction error. Watermark pixel value is calculated as s = predicted value + modified prediction error. If $s < 0$ or $s > 255$ then set the location map lm=1 else 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 pixel values. Prediction error is calculated for every even column pixel as e = watermarked pixel value – predicted value. Extracted watermark bit b as b = e mod 2. Calculate the original prediction error as $e' = \left\lfloor \frac{e}{2} \right\rfloor$. 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 T2=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 T1 or T2 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 T1=10 (in Phase-I) and T2=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 = Prediction error mod 2. Original prediction error is calculated as $e = \left\lfloor \frac{\text{Prediction Error}}{2} \right\rfloor$. 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 = Prediction error mod 2. Original prediction error is calculated as e1 = $\left\lfloor \frac{\text{Prediction Error}}{2} \right\rfloor$. Now original pixel is restored by adding predicted value with e1.

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 × n

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

Output: A watermarked image of order m×n

Step 1: A cover image 'Pic' of order m × 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 ith 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_1=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_1=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_{i+1} by considering s_i as the predicted pixel value.

d) Prediction error e_i is calculated as e_i=original pixel value – 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 $b_2$is embedded into second pixel $s_{i+1}$by expanding the prediction error e_i such as e_i'=e_i+ b_2.Here e_i' is the modified prediction error.

g) Calculate a value 'v' by adding modified prediction error with original pixel value as v=s_{i+1} + e_i'

h) If v > 255 or v < 0 then overflow or underflow occurred and set the location map value as lmap(i)=2 and reject this ith 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()" 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 ith 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 er_i is calculated as er_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 $|er_i|$ > T2 then lm(i)=2 and continue from Step 8 for next sub-block. Here T2 is the threshold value for Phase-II.

d) Consider a watermark bit say p_1 has to be embedded into first pixel s_i' by expanding the prediction error er_i such as er_i'=er_i + p_1. Here er_i' is the modified prediction error.

e) Calculate a value 'm' by adding modified prediction error with original pixel value as m=s_i' + er_i'

f) If m> 255 or m< 0 then overflow or underflow occurs and set the location map value as lm(i)=2 and reject this ith sub-block from data embedding and continue from Step 8.

g) 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 ($s_i´´$, $s_{i+1}´´$) from Phase-II. Here $s_{i+1}´´=s_{i+1}$.

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.

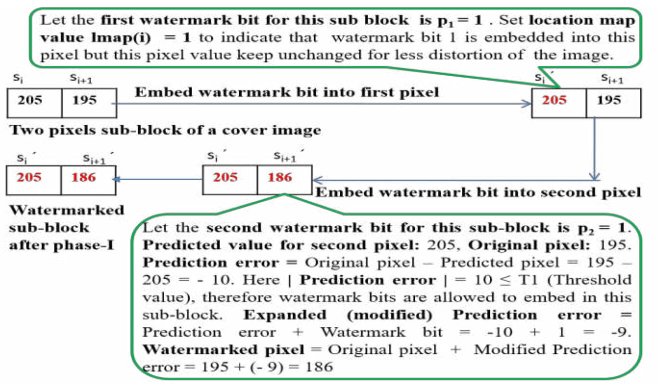
205	195	147	150
240	250	185	185

Fig 1: Cover image of order 2×4

205 | 195 Sub-block: 1	147 | 150 Sub-block: 2
240 | 250 Sub-block: 3	185 | 185 Sub-block: 4

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

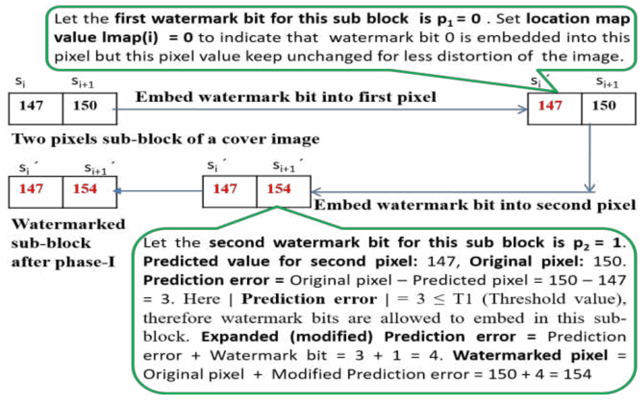
In Phase-I considering threshold value T1=10 (shown in fig. 3), I embed the secret bits in different sub-block. For example, in Fig. 3(a), a watermark bit says '1' is inserted into the first pixel of the sub-block (Sub-block-1) by setting the location map value lmap(i)=1, here i is the sub-block number. 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| ≤ threshold value T1(=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) watermarked 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 lmap(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.


s_i
s_{i+1}
205
195
Embed watermark bit into first pixel
Two pixels sub-block of a cover image
$s_i´$
$s_{i+1}´$
205
186
Embed watermark bit into second pixel
Watermarked sub-block after phase-I

Let the **first watermark bit for this sub block is p_1 = 1** . Set **location map value lmap(i) = 1** to indicate that watermark bit 1 is embedded into this pixel but this pixel value keep unchanged for less distortion of the image.

Let the **second watermark bit for this sub-block is p_2 = 1**. **Predicted value for second pixel:** 205, **Original pixel:** 195. **Prediction error** = Original pixel – Predicted pixel = 195 – 205 = - 10. Here | **Prediction error** | = 10 ≤ T1 (Threshold value), therefore watermark bits are allowed to embed in this sub-block. **Expanded (modified) Prediction error** = Prediction error + Watermark bit = -10 + 1 = -9. **Watermarked pixel** = Original pixel + Modified Prediction error = 195 + (- 9) = 186

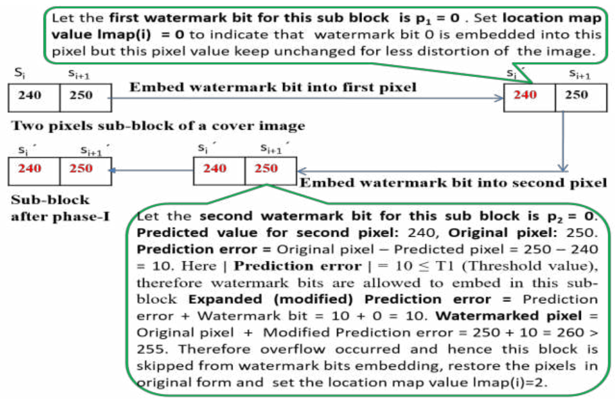
Fig: 3 (a)


s_i
s_{i+1}
147
150
Embed watermark bit into first pixel
Two pixels sub-block of a cover image
$s_i´$
$s_{i+1}´$
147
154
Embed watermark bit into second pixel
Watermarked sub-block after phase-I

Let the **first watermark bit for this sub block is p_1 = 0** . Set **location map value lmap(i) = 0** to indicate that watermark bit 0 is embedded into this pixel but this pixel value keep unchanged for less distortion of the image.

Let the **second watermark bit for this sub block is p_2 = 1**. **Predicted value for second pixel:** 147, **Original pixel:** 150. **Prediction error** = Original pixel – Predicted pixel = 150 – 147 = 3. Here | **Prediction error** | = 3 ≤ T1 (Threshold value), therefore watermark bits are allowed to embed in this sub-block. **Expanded (modified) Prediction error** = Prediction error + Watermark bit = 3 + 1 = 4. **Watermarked pixel** = Original pixel + Modified Prediction error = 150 + 4 = 154

Fig: 3 (b)


s_i
s_{i+1}
240
250
Embed watermark bit into first pixel
Two pixels sub-block of a cover image
$s_i´$
$s_{i+1}´$
240
250
Embed watermark bit into second pixel
Sub-block after phase-I

Let the **first watermark bit for this sub block is p_1 = 0** . Set **location map value lmap(i) = 0** to indicate that watermark bit 0 is embedded into this pixel but this pixel value keep unchanged for less distortion of the image.

Let the **second watermark bit for this sub block is p_2 = 0**. **Predicted value for second pixel:** 240, **Original pixel:** 250. **Prediction error** = Original pixel – Predicted pixel = 250 – 240 = 10. Here | **Prediction error** | = 10 ≤ T1 (Threshold value), therefore watermark bits are allowed to embed in this sub-block **Expanded (modified) Prediction error** = Prediction error + Watermark bit = 10 + 0 = 10. **Watermarked pixel** = Original pixel + Modified Prediction error = 250 + 10 = 260 > 255. Therefore overflow occurred and hence this block is skipped from watermark bits embedding, restore the pixels in original form and set the location map value lmap(i)=2.

Fig: 3 (c)

Fig 3: Watermark bits embedding in Phase-I taking threshold value T1=10

In Phase-II (shown in figure 4) for every sub-block, I embed the secret bits by using threshold value T2=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| ≤ threshold value T2 (=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 lm(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 T2=10 therefore skip this sub-block from data embedding for Phase-II and set the location map value lm(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.

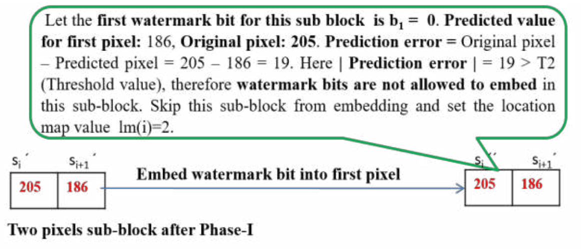
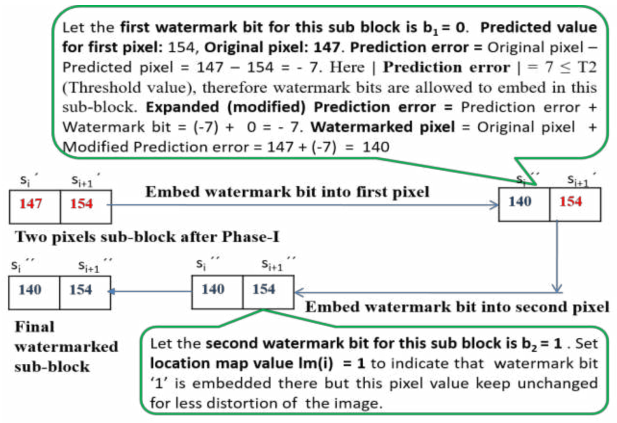
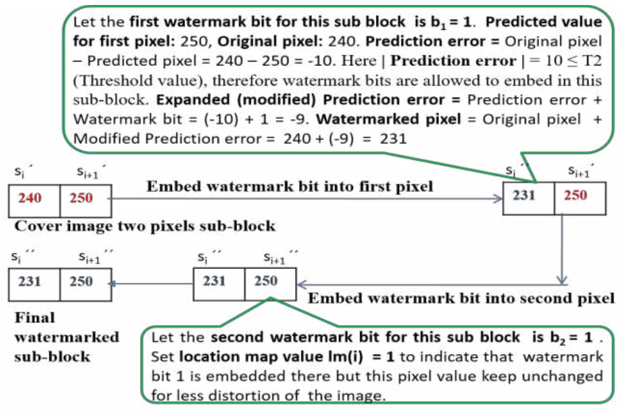

Let the **first watermark bit for this sub block is b_1 = 0. Predicted value for first pixel:** 186, **Original pixel: 205. Prediction error** = Original pixel – Predicted pixel = 205 – 186 = 19. Here | **Prediction error** | = 19 > T2 (Threshold value), therefore **watermark bits are not allowed to embed** in this sub-block. Skip this sub-block from embedding and set the location map value lm(i)=2.
s_i'
s_{i+1}'
205
186
Embed watermark bit into first pixel
s_i''
s_{i+1}''
205
186
Two pixels sub-block after Phase-I

Fig. 4(a)


Let the **first watermark bit for this sub block is b_1 = 0. Predicted value for first pixel:** 154, **Original pixel: 147. Prediction error** = Original pixel – Predicted pixel = 147 – 154 = - 7. Here | **Prediction error** | = 7 ≤ T2 (Threshold value), therefore watermark bits are allowed to embed in this sub-block. **Expanded (modified) Prediction error** = Prediction error + Watermark bit = (-7) + 0 = - 7. **Watermarked pixel** = Original pixel + Modified Prediction error = 147 + (-7) = 140

Let the **second watermark bit for this sub block is b_2 = 1** . Set **location map value lm(i) = 1** to indicate that watermark bit ‘1’ is embedded there but this pixel value keep unchanged for less distortion of the image.
s_i'
s_{i+1}'
147
154
Embed watermark bit into first pixel
s_i''
s_{i+1}'
140
154
Two pixels sub-block after Phase-I
s_i''
s_{i+1}''
140
154
s_i''
s_{i+1}'
140
154
Embed watermark bit into second pixel
Final watermarked sub-block

Fig. 4(b)


Let the **first watermark bit for this sub block is b_1 = 1. Predicted value for first pixel:** 250, **Original pixel:** 240. **Prediction error** = Original pixel – Predicted pixel = 240 – 250 = -10. Here | **Prediction error** | = 10 ≤ T2 (Threshold value), therefore watermark bits are allowed to embed in this sub-block. **Expanded (modified) Prediction error** = Prediction error + Watermark bit = (-10) + 1 = -9. **Watermarked pixel** = Original pixel + Modified Prediction error = 240 + (-9) = 231

Let the **second watermark bit for this sub block is b_2 = 1** . Set **location map value lm(i) = 1** to indicate that watermark bit 1 is embedded there but this pixel value keep unchanged for less distortion of the image.
s_i'
s_{i+1}'
240
250
Embed watermark bit into first pixel
s_i'
s_{i+1}'
231
250
Cover image two pixels sub-block
s_i''
s_{i+1}''
231
250
s_i''
s_{i+1}''
231
250
Embed watermark bit into second pixel
Final watermarked sub-block

Fig. 4(c)

Fig 4: Watermark bits embedding in Phase-II taking threshold value T2=10

Data Extraction Procedure

Algorithm for data extraction and image recovery

Input: A watermark image Pic´ of order m × 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 ith 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 lm(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 = | Prediction error mod 2 | = | er_i' mod 2 |.

d) Original prediction error is calculated as $er_i'' = \left\lfloor \frac{\text{Prediction Error}}{2} \right\rfloor = \left\lfloor \frac{er_i'}{2} \right\rfloor$. 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 lm(i)=1 then the extracted bit is p_2=1 else if lm(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 ith sub-block as (s_i', s_{i+1}').

b) If lmap(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 lmap(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 $s_i = s_i{}'$

d) Now extract the watermark bit from second pixel $s_{i+1}{}'$. Here the second pixel $s_{i+1}{}'$ is the original pixel value and the predicted pixel value is $s_i{}'$. 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 $e_i{}' = s_{i+1}{}' - s_i{}'$. Extracted bit from second pixel is $b_2 = |\text{Prediction error mod } 2| = |e_i{}' \text{ mod } 2|$.

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

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

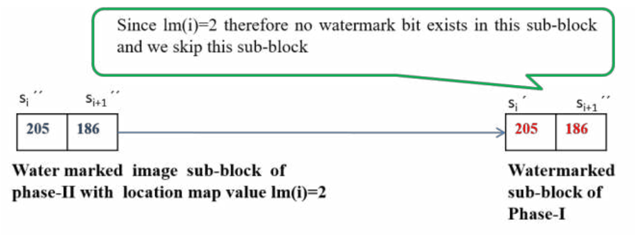
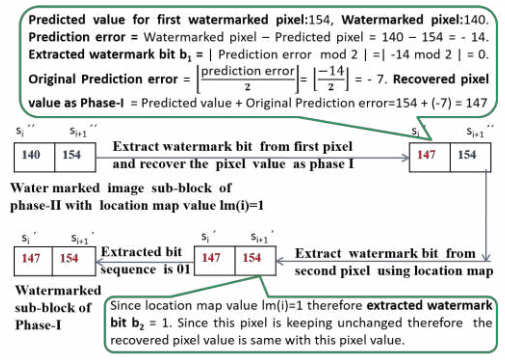
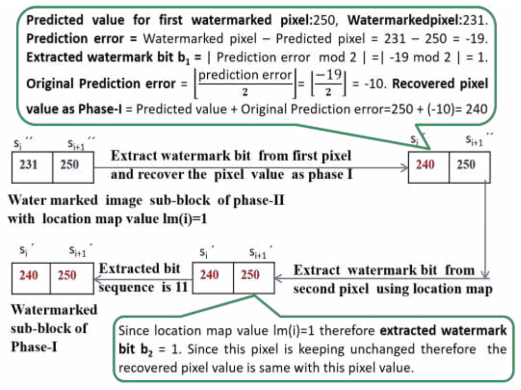

Since lm(i)=2 therefore no watermark bit exists in this sub-block and we skip this sub-block
$s_i{}''$
$s_{i+1}{}''$
205
186
$s_i{}'$
$s_{i+1}{}''$
205
186
Water marked image sub-block of phase-II with location map value lm(i)=2
Watermarked sub-block of Phase-I

Fig. 5(a)


Predicted value for first watermarked pixel:154, Watermarked pixel:140. Prediction error = Watermarked pixel – Predicted pixel = 140 – 154 = - 14. Extracted watermark bit b_1 = | Prediction error mod 2 | =| -14 mod 2 | = 0. Original Prediction error = $\left\lfloor \frac{\text{prediction error}}{2} \right\rfloor = \left\lfloor \frac{-14}{2} \right\rfloor$ = - 7. Recovered pixel value as Phase-I = Predicted value + Original Prediction error=154 + (-7) = 147

Since location map value lm(i)=1 therefore extracted watermark bit b_2 = 1. Since this pixel is keeping unchanged therefore the recovered pixel value is same with this pixel value.
$s_i{}''$
$s_{i+1}{}''$
140
154
Extract watermark bit from first pixel and recover the pixel value as phase I
$s_i{}'$
$s_{i+1}{}''$
147
154
Water marked image sub-block of phase-II with location map value lm(i)=1
$s_i{}'$
$s_{i+1}{}'$
147
154
Extracted bit sequence is 01
147
154
Extract watermark bit from second pixel using location map
Watermarked sub-block of Phase-I

Fig. 5(b)


Predicted value for first watermarked pixel:250, Watermarkedpixel:231. Prediction error = Watermarked pixel – Predicted pixel = 231 – 250 = -19. Extracted watermark bit b_1 = | Prediction error mod 2 | =| -19 mod 2 | = 1. Original Prediction error = $\left\lfloor \frac{\text{prediction error}}{2} \right\rfloor = \left\lfloor \frac{-19}{2} \right\rfloor$ = -10. Recovered pixel value as Phase-I = Predicted value + Original Prediction error=250 + (-10)= 240

Since location map value lm(i)=1 therefore extracted watermark bit b_2 = 1. Since this pixel is keeping unchanged therefore the recovered pixel value is same with this pixel value.
$s_i{}''$
$s_{i+1}{}''$
231
250
Extract watermark bit from first pixel and recover the pixel value as phase I
$s_i{}'$
$s_{i+1}{}''$
240
250
Water marked image sub-block of phase-II with location map value lm(i)=1
$s_i{}'$
$s_{i+1}{}'$
240
250
Extracted bit sequence is 11
240
250
Extract watermark bit from second pixel using location map
Watermarked sub-block of Phase-I

Fig. 5(c)

Fig 5: Extract watermark bits embedded in Phase-II and recover the pixels in the form which was exists as in Phase-I.

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.

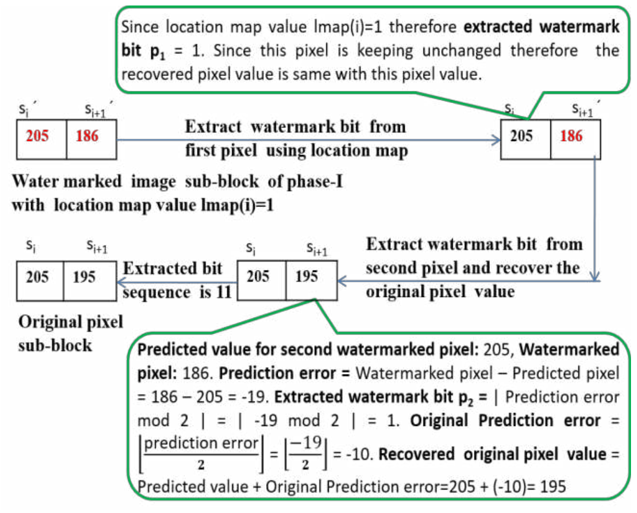
Prediction error is calculated for this sub-block by considering first pixel $s_i{}''$=140 as watermarked pixels and second pixel $s_{i+1}{}''$=154 as predicted pixel i.e. prediction error $er_i{}'$ = watermarked pixel – predicted pixel = $s_i{}'' - s_{i+1}{}''$ = 140 - 154 = -14. Extracted bit from first pixel is b_1 = | Prediction error mod 2| = | -14 mod 2| = 0. Original prediction error is calculated as $er_i{}'' = \left\lfloor \frac{\text{Prediction Error}}{2} \right\rfloor = \left\lfloor \frac{er_i{}'}{2} \right\rfloor = \left\lfloor \frac{-14}{2} \right\rfloor = -7.$

Now watermarked pixel of Phase-I is restored by adding predicted value with e i.e. Recovered pixel as in Phase-I $s_i{}'$ = predicted pixel + original prediction error= $s_{i+1}{}'' + er_i{}''$= 154 + (-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 therefor recovered pixel as in Phase-I 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 "lmap" 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 lmap(i)=2 therefore this sub-block (Sub-block: 3) not contains any watermark bits and skip it from extracting.

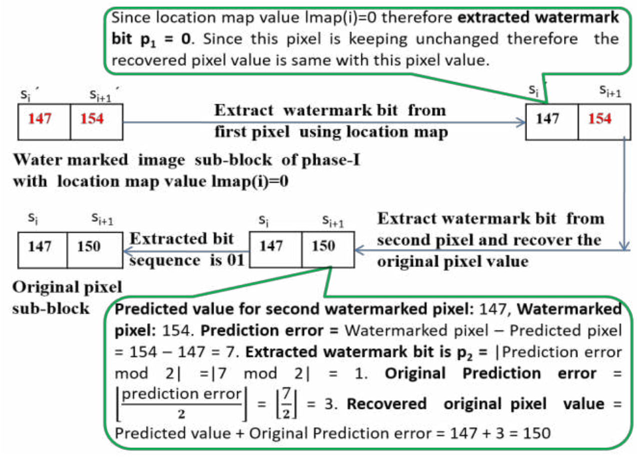
In figure 6(a) since the location map value lmap(i)=1 therefore this sub-block (Sub-block: 1) contains two watermark bits. Now extract the first watermark bit from first pixel $s_i{}'$ as since lmap(i)=1 therefore the extracted watermark bit is p_1=1. Since first pixel is kept unchanged therefor recovered original pixel is s_i = 205. Now second watermark bit is extracted from second pixel as follows:


s_i' s_{i+1}'
205 186
Extract watermark bit from first pixel using location map
s_i s_{i+1}
205 186
Water marked image sub-block of phase-I with location map value lmap(i)=1
s_i s_{i+1}
205 195
Extracted bit sequence is 11
s_i s_{i+1}
205 195
Extract watermark bit from second pixel and recover the original pixel value
Original pixel sub-block

Since location map value lmap(i)=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.

Predicted value for second watermarked pixel: 205, **Watermarked pixel:** 186. **Prediction error** = Watermarked pixel – Predicted pixel = 186 – 205 = -19. **Extracted watermark bit p_2** = | Prediction error mod 2 | = | -19 mod 2 | = 1. **Original Prediction error** = $\left\lfloor\frac{\text{prediction error}}{2}\right\rfloor = \left\lfloor\frac{-19}{2}\right\rfloor$ = -10. **Recovered original pixel value** = Predicted value + Original Prediction error=205 + (-10)= 195

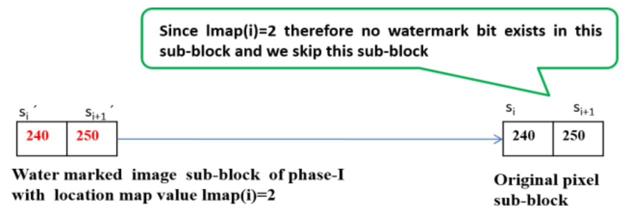
Fig. 6(a)


s_i' s_{i+1}'
147 154
Extract watermark bit from first pixel using location map
s_i s_{i+1}
147 154
Water marked image sub-block of phase-I with location map value lmap(i)=0
s_i s_{i+1}
147 150
Extracted bit sequence is 01
s_i s_{i+1}
147 150
Extract watermark bit from second pixel and recover the original pixel value
Original pixel sub-block

Since location map value lmap(i)=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.

Predicted value for second watermarked pixel: 147, **Watermarked pixel:** 154. **Prediction error** = Watermarked pixel – Predicted pixel = 154 – 147 = 7. **Extracted watermark bit is p_2** = |Prediction error mod 2| =|7 mod 2| = 1. **Original Prediction error** = $\left\lfloor\frac{\text{prediction error}}{2}\right\rfloor = \left\lfloor\frac{7}{2}\right\rfloor$ = 3. **Recovered original pixel value** = Predicted value + Original Prediction error = 147 + 3 = 150

Fig. 6(b)


s_i' s_{i+1}'
240 250
s_i s_{i+1}
240 250
Water marked image sub-block of phase-I with location map value lmap(i)=2
Original pixel sub-block

Since lmap(i)=2 therefore no watermark bit exists in this sub-block and we skip this sub-block

Fig. 6(c)

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_{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 = | Prediction error mod 2| =. | -19 mod 2| = 1. Original prediction error is calculated as $e_i'' = \left\lfloor\frac{\text{Prediction Error}}{2}\right\rfloor = \left\lfloor\frac{e_i'}{2}\right\rfloor = \left\lfloor\frac{-19}{2}\right\rfloor = -10.$ Now original pixel is restored by adding predicted value with e_i'' i.e. Recovered original pixel s_{i+1}= =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],1,1).

In the proposed method each 2 pixels' sub-block could contain maximum two bits. Therefore, the total number of watermark bits could be embedded in Phase-I into a cover image of order m × n is (m × n) $\times\frac{2}{2}$=(m × n). 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 × 512) × $\frac{2}{2}$= 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 $\log_{10}\frac{255^2}{MSE}$ ≥ 10 $\log_{10}\frac{255^2\times 2}{2}$= 48.13dB.

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)

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 130560bits, 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 and131267.55 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 1: Comparisons of experimental results (Embedding Capacity) of proposed method (taking threshold value T1=10 and T2=10) with other method [20,21]

		Jaiswal et al’s method [20]	**Kumar et al’s metho d [21]**	**The proposed method**
Cover Images	**Phases**	**EC**	**EC**	**EC**
Airplane	Phase-I	239325	130560	226800
	Phase-II	243378	261120	421510
Barbara	Phase-I	211843	130560	200098
	Phase-II	225116	261120	365692
Boat	Phase-I	207974	130560	210854
	Phase-II	238631	261120	370650
Goldhill	Phase-I	178190	130560	204084
	Phase-II	195274	261120	344912
Lena	Phase-I	223545	130560	234616
	Phase-II	236375	261120	423906
Pepper	Phase-I	201132	130560	225476
	Phase-II	224856	261120	383396
Sailboat	Phase-I	200319	130560	205250
	Phase-II	216638	261120	367352
Tiffany	Phase-I	199194	130560	221338
	Phase-II	211544	261120	401052
Zelda	Phase-I	216320	130560	247872
	Phase-II	233840	261120	453018
Average value	**Phase-I**	**208649.11**	**130560**	**219598.67**
	Phase-II	**225072.44**	**261120**	**392387.55**

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

		Jaiswal et al's method [20]	**Kumar et al's method [21]**	**The proposed method**
Cover Images	**Phases**	**PSNR**	**PSNR**	**PSNR**
Airplane	Phase-I	33.91	34.56	40.76
	Phase-II	32.81	33.07	37.05
Barbara	Phase-I	33.49	33.89	40.74
	Phase-II	31.50	31.97	37.57
Boat	Phase-I	36.36	36.10	39.22
	Phase-II	31.70	32.02	36.15
Goldhill	Phase-I	35.86	35.56	38.48
	Phase-II	34.60	34.89	35.68
Lena	Phase-I	36.58	36.25	39.52
	Phase-II	33.85	33.45	35.84
Pepper	Phase-I	36.45	36.86	38.20
	Phase-II	34.47	34.89	35.23
Sailboat	Phase-I	32.19	32.69	39.90
	Phase-II	30.46	30.85	36.79
Tiffany	Phase-I	34.86	35.01	39.86
	Phase-II	33.49	33.90	36.33
Zelda	Phase-I	37.91	37.39	39.63
	Phase-II	36.73	36.44	35.60
Average value	**Phase-I**	**35.29**	**35.37**	**39.59**
	Phase-II	**33.29**	**33.50**	**36.3**

PSNR: Peak signal to noise ratio.

Comparison Graphs

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

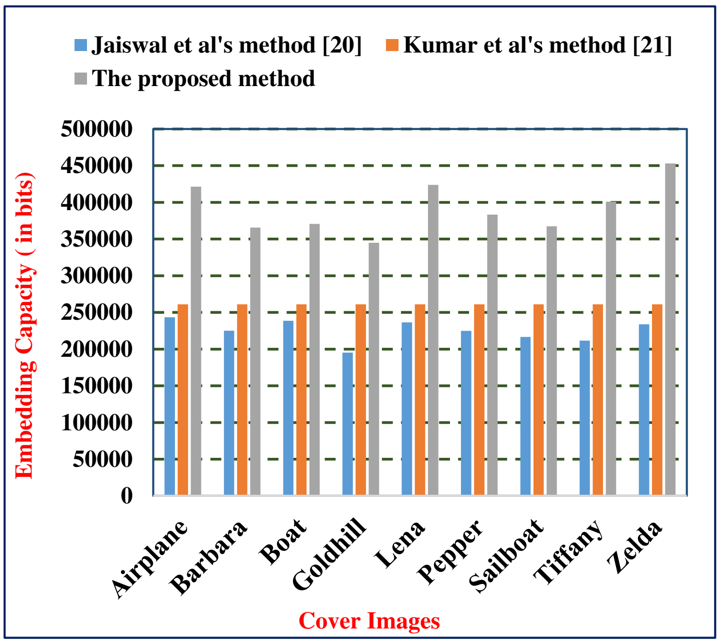


Fig 9: Graph to show a comparison of embedding capacity of proposed scheme with existing schemes (Phase-II result).

The PSNR comparisons Graph is shown below in Fig 10.

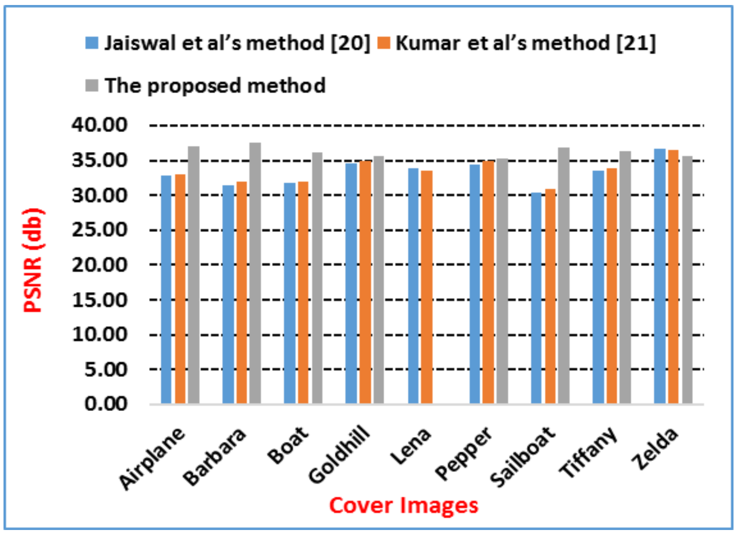


Fig 10: Graph to show a comparison of PSNR results of proposed scheme with existing schemes (Phase-II result).

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

[1] Horng SJ, Rosiyadi D, Fan P, Wang X, Khan MK (2013) An adaptive watermarking scheme for egovernment document images. Multimed Tool Appl 72(3):3085–3103

[2] Sharma A, Singh AK, Kumar P (2017) Combining Haar wavelet and KarhunenLoeve transforms for robust and imperceptible data hiding using digital images. J Intell Syst. https://doi.org/10.1515/jisys-20170032

[3] Cox IJ, Miller ML, Bloom JA, Fridrich J, Kalker T. Digital Watermarking and Steganography, 2nd edn. Morgan Kaufmann: U.S.A., 2008.

[4] Feng JB, Lin IC, Tsai CS, Chu YP. Reversible watermarking: current status and key issues. International Journal of Network Security 2006; 2(3): 161–171.

[5] W. Hong, T.S. Chen, C.W. Shiu, Reversible data hiding for high quality images using modification of prediction errors, Journal of Systems and Software 82 (11) (2009) 1833–1842.

[6] J. Fridrich, M. Goljan, R. Du, Lossless data embedding - new paradigm in digital watermarking, EURASIP Journal on Applied Signal Processing 2002 (2) (2002) 185–196.

[7] M.U. Celik, G. Sharma, A.M. Tekalp, E. Saber, Lossless generalized- LSB data embedding, IEEE Transactions on Image Processing 14 (2) (2005) 253–266

[8] Tian, J. Reversible data embedding using a difference expansion. IEEE Transactions on Circuits and Systems for Video Technology 13 (2003) 890-896.

[9] Alattar, A.M. Reversible watermark using the difference expansion of a generalized integer transform. IEEE Transactions on Image Processing 13 (2004) 1147-1156.

[10] Tai, W.L., Yeh, C.M. and Chang, C.C. Reversible data hiding based on histogram modification of pixel differences. IEEE Transactions on Circuits and Systems for Video Technology 19 (2009) 906-910.

[11] Al-Qershi, O.M. and Khoo, B.E. Two-dimensional difference expansion (2D-DE) scheme with a characteristics-based threshold. Signal Processing 93 (1) (2013) 154-162.

[12] Ni Z, Shi Y, Ansari N, Su W. Reversible data hiding. IEEE Transactions on Circuits and Systems for Video Technology 2006; 16(3): 354–362.

[13] Zhao, Z., Luo, H., Lu, Z.M. and Pan, J.S. Reversible data hiding based on multilevel histogram modification and sequential recovery. AEU -International Journal of Electronics and Communications 65 (2011) 814-826.

[14] Huang, L.C., Tseng, L.Y. and Hwang, M.S. A reversible data hiding method by histogram shifting in high quality medical images. The Journal of Systems and Software 86 (2013) 716-727

[15] Luo, H., Yu, F.X., Chen, H., Huang, Z.L., Li, H. and Wang, P.H. Reversible data hiding based on block median preservation. Information Sciences 181 (2011) 308-328.

[16] Thodi DM, Rodrguez JJ. Expansion embedding techniques for reversible watermarking. IEEE Transactions on Image Processing 2007; 16(3): 721–730.

[17] Y. Hu, H.K. Lee, J. Li, DE-based reversible data hiding with improved overflow location map, IEEE Transactions on Circuits and Systems for Video Technology 19 (2) (2009) 250–260.

[18] Li, X., Li, J., Li, B. and Yang, B. Hsiao, J.Y. High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. Signal Processing 93(2013) 198-205.

[19] S. K. Lee, Y. H. Suh and Y. S. Ho, "Reversible Image Authentication Based on Watermarking" , Multimedia and Expo, 2006 IEEE International Conference on. pp. 1321-1324, 2006.

[20] Jaiswal Sp, Oscar Au, Jakhetiya V, Guo Ay, Tiwari Ak. Adaptive Predictor Structure Based Interpolation for Reversible Data Hiding. In *Proceedings of International Workshop on Digital-forensics and Watermarking (IWDW):* Taipei, Taiwan, 2014: 276 - 288.

[21] Kumar Manoj, Agrawal Smita. Reversible Data Hiding Based on Prediction Error Expansion using Adjacent Pixels. *Security and Communication Network – Wiley Online Library* 2016; 9: 3703 -3712.