iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** V **Month of publication:** May 2020

DOI: http://doi.org/10.22214/ijraset.2020.5121

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

Watermarking using Weighted Prediction

K. Naveen Kumar[1], Nagulancha Raju[2], Gondhi Navabharat Reddy[3], Gondhi Vikram Reddy[4]
[1, 2]*Assistant Professor ECE, Vignan institute of science and Technology*
[3, 4]*Assistant Professor ECE, Electrical Engineer, Vignan institute of science and Technology*

Abstract: ***Reversible digital watermarking techniques enable the recovery of the original 'cover image' from a watermarked image in a distortion-free way. Reversible watermarking techniques find application in military and medical imagery, where integrity of the cover image is of utmost importance. In this study, we propose a reversible digital image watermarking algorithm that predicts a pixel grey scale value exploiting its correlation with its neighbouring pixels, and embeds watermark bits into the prediction errors. Our algorithm succeeds in providing high embedding capacity with very low distortion, without 'multilayer embedding', hence reducing the computational burden compared with existing state-of-the- art algorithms. In addition, our results indicate that the resulting data overhead requirement is extremely low.***
Keywords: cover image, distortion-fre, pixel

I. INTRODUCTION

Watermarking embeds information into a digital signal like audio, image, or video. Reversible watermarking is the process of hiding information within a multimedia data (images, audio or video), for the purpose of content protection and authentication. In digital watermarking the secret information which is generally in the form of bits which is known as 'watermark' is embedded in to an image (cover image) in such a way that the distortion in over image due to watermarking is almost negligible perceptually. Reversible image watermarking can restore the original image without any distortion after the hidden data is extracted, which means the image restored after extraction of watermark is identical to the original cover image, pixel by pixel, bit by The reversible watermarking has wide applications in military and medical imagery, where distortion free recovery of original image after watermark extraction has primary importance In common methods for reversible watermarking of digital images, the high spatial correlation between neighbouring pixels is used for watermarking. Any feature of cover image is selected and then it is modified for embedding the watermark bits. For example, the greyscale values of pixels, the difference of adjacent pixels' greyscale values and the quantisation or interpolation errors after the pixels are quantised or interpolated, respectively, are some of the features selected by various investigators. Most of the reversible watermarking algorithms perform 'multi-layer watermark embedding', by iteratively watermarking an already watermarked image so as to achieve high embedding capacity.

In this paper, we introduced reversible watermarking to greyscale images. Our algorithm exploits the spatial correlation among the neighboring pixels. For predicting the value of common neighbouring pixels we considered median of agroup of pixels. To enhance the accuracy of prediction we assigned weights to the neighbouring pixels during the prediction, then a 'weighted median' is calculated. Difference of the pixel values with their predicted values is then modified for embedding the watermark. The two main advantages of the proposed technique are

The proposed technique gives higher embedding capacity without using multilayer embedding and required less computational effort.

1) Number of overhead bits required in algorithm is extremely low (actually zero in a majority of the benchmark images considered

II. SYSTEM MODEL

The proposed algorithm used high spatial correlation among the neighbouring pixel values in grey scale images. It predicts the greyscale value of a pixel from those of its neighbouring pixels and the watermark bits are embedded into the prediction errors. Depending on the order of prediction, the pixels are divided into four classes as follows:The pixels of the cover image whose values remain unchanged during the watermarking process are termed 'base pixels'.

A. The pixels of the cover image whose values remain unchanged during the watermarking process are termed 'base pixels'.
B. From the base pixels, the 'first set of predicted pixel values' is derived
C. Further, the 'second' and 'third sets of predicted pixel values' are derived from the base pixels and first set of predicted pixel values.

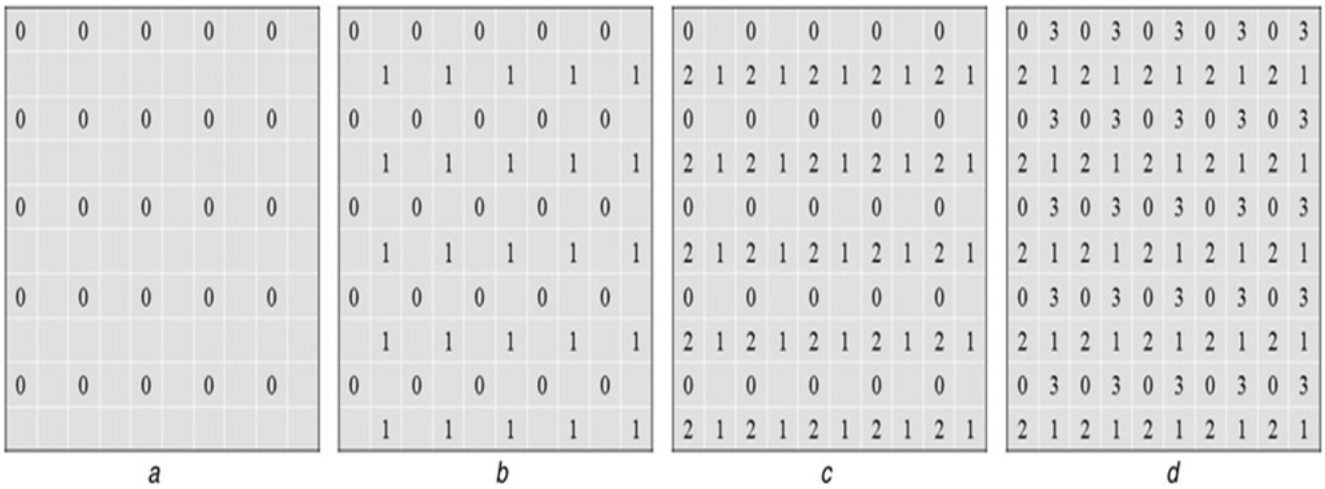


Figure 1: The location of the base pixels, first, second and third set of interpolated pixels are marked with '0', '1', '2', and, '3' in the figure.

1) Watermark Embedding Algorithm: The watermark embedding algorithm consist of mainly four steps: (i) selection of the base pixels, (ii) prediction of other pixels from the base pixels, (iii) calculating the prediction errors, (iv) embedding watermark bits in to the errors. The flow chart of the watermark embedding algorithm is shown as figure 2.

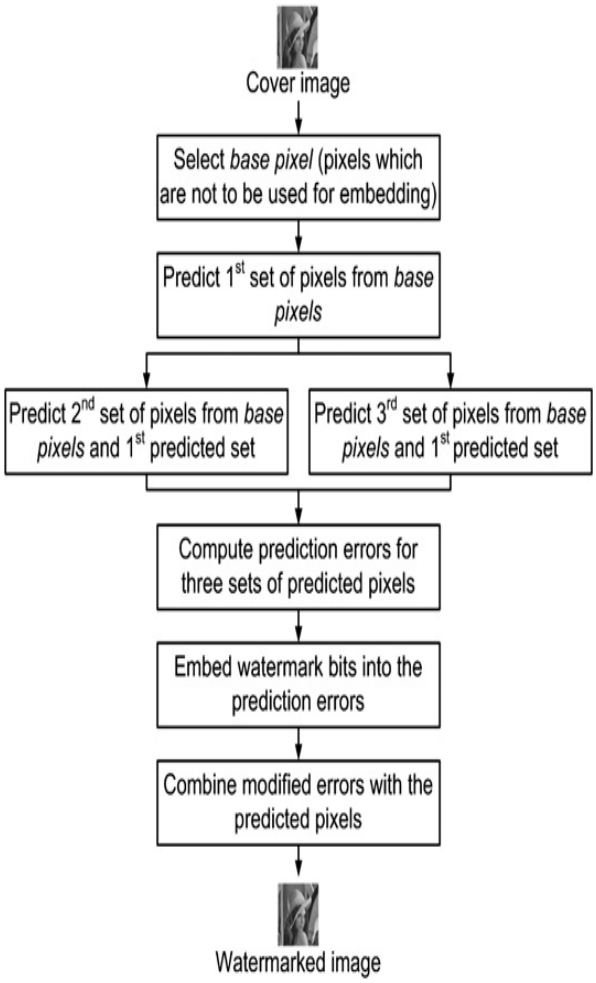


Fig. 2 Flow chart of the proposed FDBA

The steps in embedding algorithm is explained in detail as follows

a) *Selection of base Pixels:* One out of every four pixels in the cover image is taken as the base pixels in our algorithm. By taking in this method we get base pixels uniformly distributed throughout the cover image. The position of base pixels is represented as '0' in Figure 1.

b) *Prediction of Three Sets of Interpolated Pixels:* For prediction of the greyscale value of a pixel, we take the weighted median of adjacent pixels. Since adjacent pixels are usually highly correlated, the weighted median of neighbouring pixels provides good prediction for most pixels. To compute weighted median of n integer pixel values

i1 ,i2 ,i3 ,.....,in , the integer values are first sorted in to sequence

i ,i ,i ,.....,i , such that i □ i □ i , □□, □ i

. Then their median is

(1) (2) (3) (n) (1) (2) (3) (n)

calculated as

$$\text{median}(i_{(1)}, i_{(2)}, \ldots, i_{(n)}) = i_{\frac{(n+1)}{2}}$$

if n is odd

$$= \frac{i_{(n/2)} + i_{(n/2)+}}{2} \quad \text{otherwise}$$

The weighted median of n pixel values p1, p2, . . ., pn is computed as

WM {{ p1 , p2 ,....., pn },{w1 , w2 ,..., wn }} □ median(w □p , w □p ,...., w □p)

1 2 n 1 2 n 1 1 2 2 n n

Where the non-negative integers w1 , w2 ,..., wn are the weights assigned to the pixels p1 , p2 ,....., pn , respectively and w □ implies that W repetitions of pixel p . For

example, let the greyscale values four pixels be 167, 168, 240, 255 and their assigned weights be w1 = 1, w2 = 2, w3 = 3, w2 =2 respectively. The weighted median of the pixels is calculated as

c) *WM {{167,168, 240, 255},{1, 2,3, 2}}*

= median(1□167, 2□168,3□240, 2□255)

= median(167,168,168, 240, 240, 240, 255, 255)

=240

In the proposed algorithm, one pixel is predicted as the weighted median of its neighbouring pixels. Some of the neighbouring pixels may be base pixels or may be predicted one themselves. So while computing the weighted medians we need assign higher weights to the original neighbouring pixels than to the predicted neighbouring pixels so as to increase the accuracy of the prediction. By doing this we try to reduce the absolute values if prediction errors. The weights assigned to the original neighbour pixels and predicted neighbour pixels are 2:1 in our algorithm.

For calculating the first set of interpolated values we use only the base pixels. But in order to calculate the second and third set of pixels we need to consider predicted values of first set of pixels also. The predicted values of first set of pixels depend on the four neighbouring base pixels surrounding it on its four corners. Since they are all base pixels we assign equal weights to all the four pixels. The prediction formula is

d) *(p(i, j)) = WM ({ p(i □1,j -1), p(i □1,j □1), p(i □1,j □1),p(i □1,j □1)},{1,1,1,1})*

where i and j are the row and column number of the pixel which is to be predicted, respectively.

The second set of pixel prediction starts from the two base pixels and two predicted pixels from the first set of predicted pixels. The two base pixels are constitute the top and bottom positions and the two predicted first set of pixels constitute the left and right positions of the pixel whose value is to be predicted. The base pixels are assigned with higher weights since they are the original non-predicted pixels. The prediction formula for the second set of pixel values is given by

e) (p(i, j)) □ WM ({ p(i, j □1), p(i □1, j), p(i, j □1), p(i □1, j)},{1, 2,1, 2})

Similarly the predicted value of each third set pixel depends on the two base pixels at the left and right places of the pixel whose value is to be calculated and the two predicted values from the first set of predicted pixel values occupying the top and bottom positions. It is to be noted that the base pixels should give higher weights than the predicted values. The prediction formulae for the third set of the pixel values is given by

f) (p(i, j)) □ WM ({ p(i, j □1), p(i □1, j), p(i, j □1), p(i □1, j)},{2,1, 2,1})

For predicting the pixels in the border or edges of image, we assign 'zero' weights to the neighbours which are absent. The figure 1 shows the order of prediction of the pixels.

2) *Computing the Prediction Errors:* Prediction error is taken as the difference between original pixel value and the predicted pixel value. For each pixel we calculate the prediction error using the integer transformation as

e = □ (p) -p

as we said earlier we use high spatial correlation between adjacent pixels, usually the errors are small integers close to zero.

3) *Embedding watermark bits:* For embedding watermark bits the prediction errors that are close to zero are used, leading to high embedding capacity. Since the number of errors close to zero are usually large, in order to define the' closeness to zero', we introduce an 'error threshold' k ≥ 0 . The | |

pixels with prediction error e≤ only are used for embedding

k

watermark bits. The watermark bit is embedded in to an error by multiplying the error by 2 and adding the watermark bit to the obtained result. For pixels whose prediction | | error e

> k , a constant

shift of magnitude (k + 1is applied to the absolute prediction error values obtained so as to 1) avoid the overlapping with the watermarked pixels. The procedure in figure 3 is followed so as to embed watermark bits in to the prediction errors.

4) *Combining Modified Errors With The Predicted Pixels:* Each watermarked pixel is obtained by combining the predicted pixel with the modified (watermark bit embedded) prediction error. The predicted pixel is combined with modified error using the equation

Pwm = □(p) - Φ(e) = p + e - Φ(e)

Where, Pwm is the watermarked pixel. This may produces some watermarked pixels falling outside the range of greyscale values [0,255], causing an underflow (p< 0) or an overflow (Pwm<255). this has to be taken care of and handling of these is explained in later section.

Procedure 1

/* Embed watermark bits into the prediction errors */

Input: original pixel prediction error *(e)*, error threshold *(k)*

Output: Modified prediction error $\phi(e)$

1: **if** $e < 0$ **then**
2: $sgn(e) \leftarrow -1$
3: **else**
4: $sgn(e) \leftarrow +1$
5: **end if**
6: **if** $(|e| > k)$ **then**
7: /* Apply the constant shift to the absolute error value */
8: $\phi(|e|) \leftarrow |e| + (k+1)$
9: **else**
10: /* $b \in \{0,1\}$ is the next watermark bit to be embedded */
11: $\phi(|e|) \leftarrow 2 * |e| + b$
12: **end if**
13: $\phi(e) \leftarrow sgn(e) * \phi(|e|)$
14: **return** $\phi(e)$

Figure 3: procedure for embedding watermark bits into prediction errors

5) *Watermark Extraction Algorithm:* Figure 4 shows the flow chart for the watermark extraction algorithm. For extraction also we choose the base pixels in the watermarked image and predict first, second and third set of pixels, using the same procedure explained in the embedding algorithm. The positions of base pixels and the three set of predicted pixel values are the same as shown in the figure 1.

$\Phi(e) = \square(p) - p$

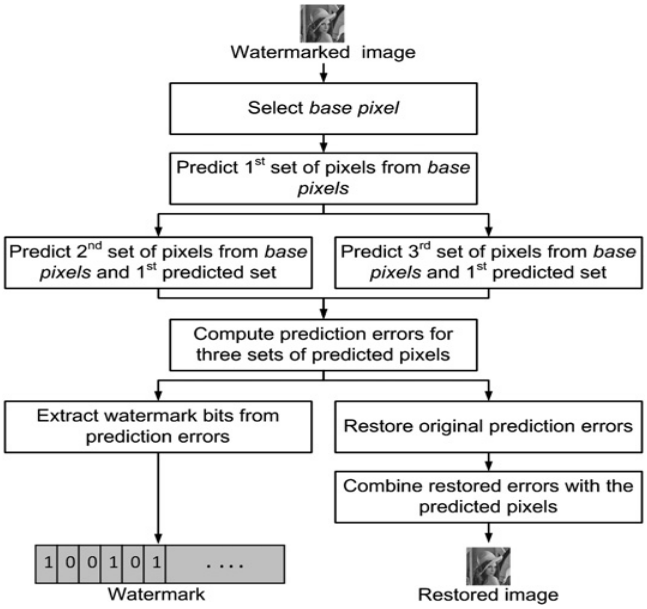


Figure 4: Watermark extraction Algorithm

Forward transformation is applied to each watermarked non-base pixel Pwm and its prediction error $\square(p)$ for the calculation of prediction error, from which the watermark bits have to be extracted. From the prediction errors ($\Phi(e)$) watermark bits are extracted and the original errors (e) are then restored through the process explained as procedure 2 in figure 5. The term 'original prediction errors' refers to the prediction pixel errors that are achieved before the watermark embedding.

Procedure 2

/* Extract watermark bits and restore original prediction errors */

Input: Watermark bit embedded prediction error ($\phi(e)$), error threshold (k)

Output: Original prediction errors (e)

1: **if** $\phi(e) < 0$ **then**

2: $sgn(\phi(e)) \leftarrow -1$

3: **else**

4: $sgn(\phi(e)) \leftarrow +1$

5: **end if**

6: **if** $|\phi(e)| > (2 * k + 1)$ **then**

7: $|e| \leftarrow |\phi(e)| - (k+1)$

8: **else**

9: /* $b \in \{0,1\}$ is the next watermark bit to be extracted */

10: $b = mod(|\phi(e)|, 2)$

11: $e \leftarrow \frac{|\phi(e)| - b}{2}$

12: **end if**

13: $e \leftarrow \text{sgn}(\phi(e)) * |e|$

14: **return** e

Figure 5: procedure for watermark extraction

We apply the reverse of $\Phi(e) = \square(p)$-Pwm transformation to each $\{ \square(Pwm), e \}$ pair in order to restore the original cover image

$p = \square(Pw) - e$

6) *Handling the overflow/underflow:* Upon embedding there is a chance that the embedded or watermarked non-base pixel can have an overflow (Pwm> 255) or an underflow(Pwm<0). We simply do not embed into a prediction error(Φ(e)), which may cause such an under/overflow, and move on to the next. During extraction we check each prediction error to find out whether it can cause an overflow or underflow. For testing this first we perform the procedure 1 (figure: 3) to prediction error with b=0 and b=1, representing the watermark bits to be embedded and then applying the transformation Pwm = □(p) - Φ(e) =p + e - Φ(e) to the modified prediction error to produce the test watermark pixel Pwm . This watermarked pixel is then tested for underflow or overflow. A prediction error, found capable of causing under/overflow, during extraction, indicates one of the two possibilities:

a) It was found to be capable of causing under/overflow during embedding, and hence was not used for embedding

b) Previously, it was capable of undergoing embedding without causing an under/overflow, so was used for embedding, but after embedding it has lost its embedding capability.

For error free extraction we need to correctly infer which of the above possibilities is actually occurred. For this we are setting threshold value, k. The value of k determines the pixels that can be embedded. We are taking the pixels with pixel values greater than 2k (p > 2k) and less than (255-2k),(p < (255 - 2k)).

7) *Avoiding Multilayer Embedding:* Depending on the size of the payload to be embedded, the need to embed greater number of bits into a cover image might arise. In such a scenario, the value of k can be increased, allowing errors with larger absolute values to be used for embedding, thus providing more space for embedding. With increase in the value of k, the amount by which the errors are modified || || (□(e) e+(k +1) is also larger, increasing the distortion.However, the higher embedding capacity comes with higher cover-image distortion in all watermarking schemes. Hence, the error threshold parameter (k) provides us with a degree of freedom. In effect, it helps us to avoid multiple layers of data embedding.

III. RESULTS AND DISCUSSION

A. *Experimental Results*

The propose algorithm was implemented in Matlab. The performance of the algorithm was tested on three 512 × 512 pixels standard image processing test images: (i) Airplane,
(ii)Lena, (iii) Mandrill. The test images were chosen to have widely varying amounts of correlation among neighbouring pixels, for example, Airplane, has high correlation, Mandrill has very low correlation. The test images are shown in the figure

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right)dB = 10\log_{10}\left(\frac{255^2}{MSE}\right)$$ very

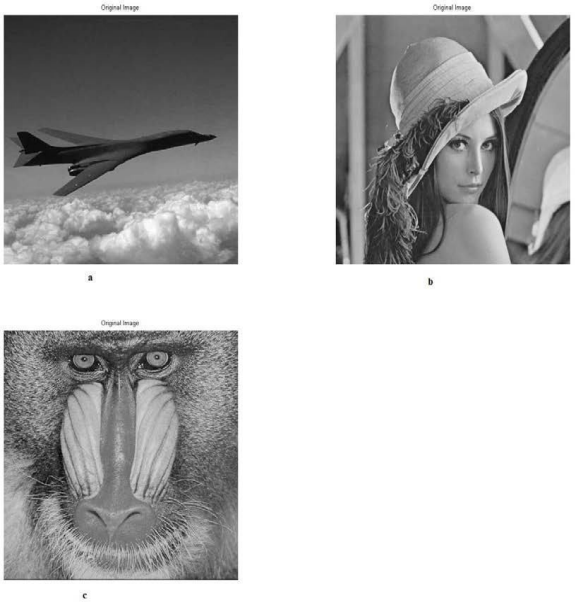


Figure 1: Standard image processing test images (a)Airplane (b) Lena, (c) Mandrill

We test the performance of the proposed algorithm with respect to the following properties,

1) Maximum embedding capacity achievable,

2) Mean Square error (MSE)

3) Distortion of watermarked image as compared with original cover image

Maximum embedding capacity of an image was evaluated by the number of pure watermark bits (not including the overhead bits) that can be embedded into the entire cover image as well as average number of bits that can be embedded per pixel, measured in units of bits per pixel. Distortion of the watermarked image was estimated in terms of peak signal-to- noise ratio (PSNR). For calculating the PSNR value first we need to calculate the Mean Square Error (MSE). MSE is calculated as

$$MSE=\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(Xorg(i,j)-Xwm(i,j))2}{M*N}$$

where Xorg (i, j) is the (i, j) th pixel of the original image, and Xwm (i, j) is the (i, j) th pixel of the watermarked image, and m and n are the dimensions of the image (here each is 512). Then, PSNR was calculated as

where MAX 1is the maximum possible pixel value of the image, which is 255 in this case because of the 8-bit greyscale nature of the image. The value of the error threshold parameter

(k) was varied from 0 to 10. The effect of this variation on the embedding capacity (of the entire cover image) in bits, PSNR (in decibels) and number of location map bits, for our three 512 × 512 greyscale standard image processing test images, has been shown in the tables 1, 2, and 3 for Airplane. Lena and Mandrill respectively

Table 1: Maximum embedding capacity, MSE and PSNR at different values of k for Airplane as cover image.

Error Threshold (k)	Maximum Capacity	MSE	PSNR
0	40442	0.5854	50.45
1	52175	0.6363	50.094
2	57249	0.66789	49.8834
3	59996	0.689	49.7483
4	61650	0.7038	49.651
5	62782	0.7143	49.5961
6	63496	0.7219	49.546
7	63766	0.7274	49.5131
8	63618	0.7303	49.496
9	63281	0.7317	49.487
10	63212	0.7327	49.481

Table 2: Maximum embedding capacity, MSE and PSNR at different values of k for Lena as cover image.

Error Threshold (k)	Maximum Capacity	MSE	PSNR
0	37201	1.11	47.6776
1	48419	1.17	47.429
2	53627	1.22	47.267
3	56700	1.2544	47.1644
4	58692	1.28	47.05
5	60042	1.3015	46.985
6	61050	1.3186	46.925
7	61830	1.3327	46.883
8	62640	1.3445	46.845
9	62947	1.35544	46.81
10	63374	1.3626	46.7871

Table 3: Maximum embedding capacity, MSE and PSNR at different values of k for Mandrill as cover image.

Error Threshold (k)	Maximum Capacity	MSE	PSNR
0	33036	6.0749	40.2594
1	35156	6.19	40.213
2	37186	6.2986	40.1386
3	39078	6.3996	40.0694
4	40836	6.4933	40.0061
5	42371	6.5814	39.9476
6	43835	6.6638	39.89
7	45136	6.7415	39.84
8	46327	6.81	39.79
9	47361	6.88	39.753
10	48394	6.94	39.71

Results presented in Tables 1, 2 and 3 shows that by increasing the value of k, very high embedding capacity can be achieved at considerably low distortion. The rate of increase of the embedding capacity, with increase in values of k, is somewhat larger for Lena compared with other images. This is owing to the previously mentioned low correlation among neighbouring pixels of Lena. The Airplane has the lowest MSE value and highest PSNR. Also as the error threshold is increased from 0 to 10 the value of PSNR is decreasing slowly for all the three images. Also from the embedded images in figure 2 we can see that even though the bits are embedded in to the cover image there are not many variations in the original and embedded image.

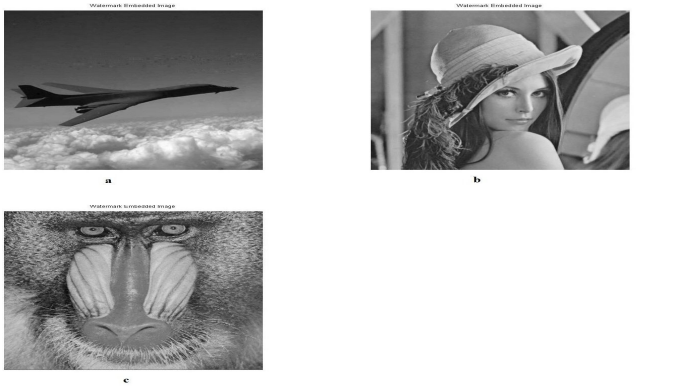


Figure 2: Watermarked Embedded images (a)Airplane (b) Lena (c) Mandrill

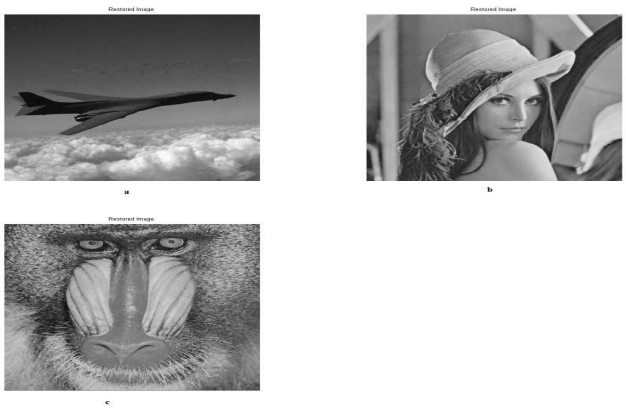


Figure 3: Restored Images (a) Airplane (b) Lena (c) Mandrill

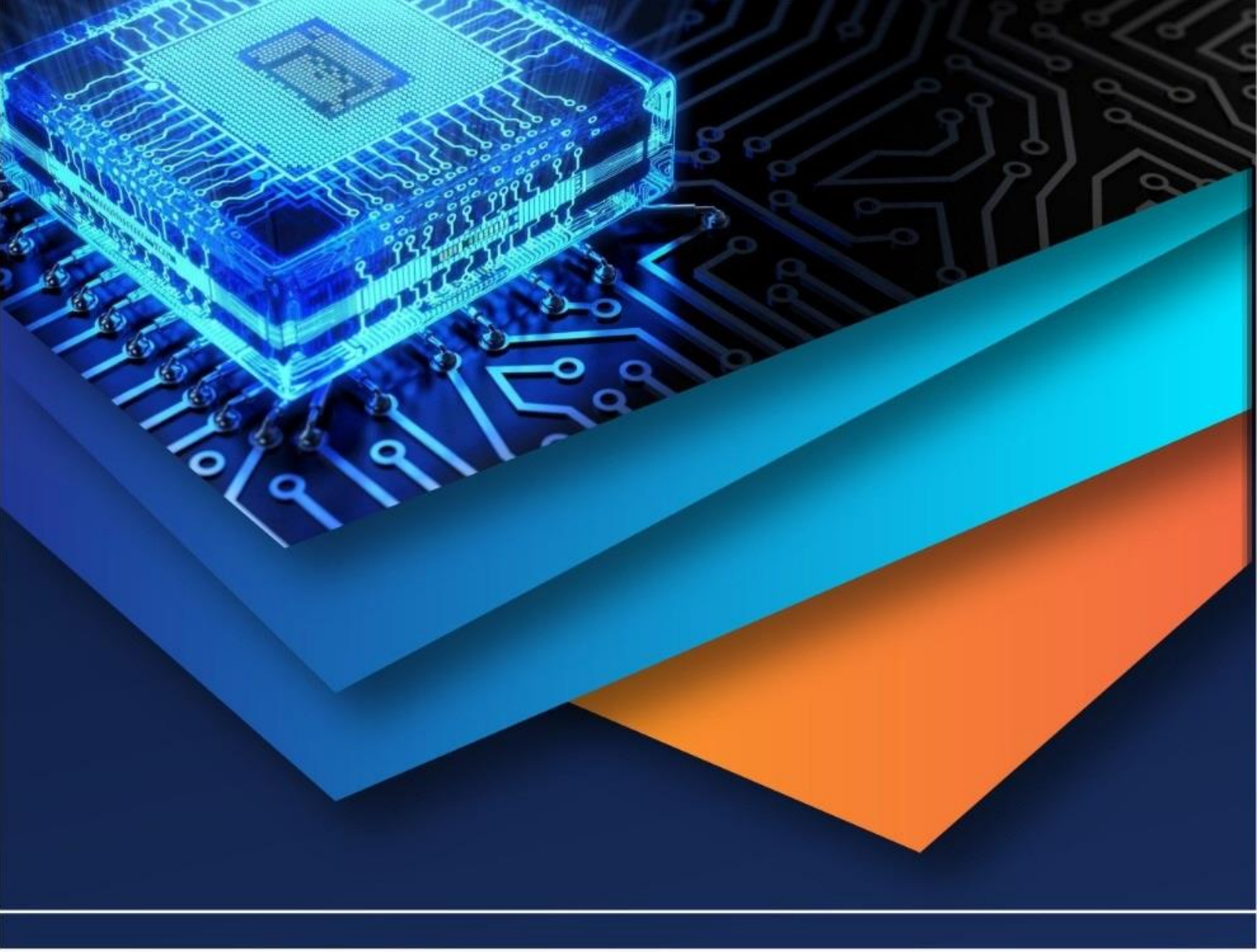
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

IV. CONCLUSION

In this study, we have presented a reversible watermarking algorithm exploiting the inherent spatial correlation among neighbouring pixel values in an image. Greyscale pixel values are predicted from the weighted median of neighbouring pixels. Assigned weights to the neighbouring pixels help to make the predictions more accurate. The proposed algorithm achieves high embedding capacity with very low distortion and lesser number of overhead bits. The computational burden of the algorithm has also been reduced by avoiding multilayer embedding. Future research would be directed towards reversible watermarking of colour images with the proposed algorithm.

REFERENCES

[1]. Shailendra Singh ,Dr. Ashok Kumar," Novel Optimal Deployment of Sensor Nodes Using Bio Inspired Algorithm" 2014 IEEE International Conference on Advanced Communication Control and Computing Technologies.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY