Reversible Watermarking Method Using Optimal Histogram Pair Shifting Based on Prediction and Sorting

Hee Joon Hwang[1], Hyoung Joong Kim[1], Vasiliy Sachnev[1] and Sang Hyun Joo[2]
[1] CIST, Graduate School of Information Management and Security, Korea University Seoul 136-701, Korea
[e-mail: {ksteiner, khj, bassvasys}@korea.ac.kr]
[2] ETRI
Daejeon 307-700, Korea
[email: joos@etri.re.kr]
*Corresponding author: Hyoung Joong Kim

Received March 31, 2010; revised June 8, 2010; accepted July 11, 2010; published August 27, 2010

Abstract

To be reversible as a data hiding method, the original content and hidden message should be completely recovered. One important objective of this approach is to achieve high embedding capacity and low distortion. Using predicted errors is very effective for increasing the embedding capacity. Sorting the predicted errors has a good influence on decreasing distortion. In this paper, we present an improved reversible data hiding scheme using upgraded histogram shifting based on sorting the predicted errors. This new scheme is characterized by the algorithm which is able to find the optimum threshold values and manage the location map effectively. Experimental results compared with other methods are presented to demonstrate the superiority of the proposed method.

Keywords: Reversible watermarking, histogram shifting, lossless data hiding, difference expansion, predicted error expansion

A preliminary version of this paper is accepted in ICONI/APIC-IST 2009, December 17-21, NUSA DUA (Bali), Indonesia. This version includes a concrete analysis and additional implementation results which support proposed reversible watermarking idea. This research was in part supported by the research grant from the IT R&D program of MKE/IITA, [2007-S-001-01, Development of Anonymity-based u-Knowledge Technology Program] and the FP7 3DLife project by the Ministry of Education, Science and Technology. We express our thanks to Dr. Richard Berke who checked our manuscript.

DOI: 10.3837/tiis.2010.08.012

1. Introduction

Reversible data hiding or lossless data hiding techniques hide data into digital content such as text, image, audio, or video, and recover not only the hidden data but also original content perfectly. This lossless data hiding technique can be applied to some applications which need perfect integrity of the original content: for example, medical and military applications. In these applications, it is deriable to utilize the reversible data hiding technique requires high embedding capacity and low distortion. Of course, these two requirements conflict each other. However, advanced techniques can hide more data with less distortion.

Difference expansion method [1] has opened a mathematical way to hide large amount of data systematically. This method takes two pixels as a pair, computes the mean and difference values of the pair, expands the difference value, and finally hides a single bit into the pair. By multiplying the difference value by 2, the resultant difference value becomes an even number. Thus, we can hide a single bit into the pair. Original pixel values can move farther from the mean value. If at least one value goes beyond the boundary of the valid range (i.e., [0, 255] for grayscale image), the pair becomes invalid due to overflow/underflow. Thus, not all pairs can be expandable. Location map marks which pairs are expanded and which pairs not. Without the location map, the maximum possible embedding capacity is half of the image size (i.e., 0.5 bits per pixel (bpp)). However, the size of the location map is also half of the image size. Then, actaully we can not hide anything into the image. Fortunately, the location map which is a binary data can be compressed using a lossless compression algorithm such as JBIG2. As a conclusion, its maximum possible embedding capacity has to be below 0.5 bpp.

Thus, one of many homeworks is to reduce the size of the location map. Kim et al. [2] has reduced the location map size by marking the ambiguous pairs only since the unambiguous pairs are obviously identified. They utilize a threshold value to tell the unambiguous region from the ambiguous region. In addition, embedding capacity can be controled by choosing appropriate threshold values. However, afterward, many methods of which location map is negligible have been invented.

Kamstra and Heijmans [3] have presented a new approach utilizing a sorting method to exploit the correlation among pairs of pixels. They form a group with four pixels rather than two and the average value of them is used as a predicted value. The prediction error is the difference value between the average value and the pixel value in the center of the surrounding four pixels(i.e., in the north, south, east, and west position). They observe that the predicted errors are highly correlated with average values of the groups. Thus, they attempt to sort the pairs by average values, and start to seek candidate pairs for data embedding by the sorted order. Since the average values are unchanged even after data hiding, the original order of groups can be recovered at the decoder. Due to the sorted groups, we are highly likely to choose the pairs with small difference values. Hiding data into the small difference values can reduce the distortion. In addition, it facilitates efficient compression of the location map, which also reduces the location map size further.

The original idea of histogram shifting has been proposed by Thodi et al. [4]. This scheme exploits the inherent vacancies or intentionally generated vacancies in the histogram to hide data. However, innate vacancies in the histogram are rare in nature. In addition, even though there are such vacancies, since their neighboring bins are small in population, embedding capacity is negligible. Thus, we pick a group of bins with large population and their neighbor bins are shifted to secure vacancies intentionally.

One of the significant contibutions of Thodi and Rodriguez [4] is the combination of the

difference expansion method with the histogram shifting method. This approach incorporates the positive and negative threshold values. Within the threshold values, the prediction errors are expanded by the difference expansion method. On the other hand, the predicted errors are shifted when the predicted errors are larger than the threshold values in magnitude. Therefore, embedding capacity is controlled by the threshold values. Recently, Hong et al. [5] have improved the work of Thodi and Rodriguez [4].

Sachnev et al. [6] applied the same idea to the predicted errors obtained using a diamond pattern. Their idea of using two kinds of non-overlapping groups of pixels is utilized in this paper. In addition, Xuan et al. [7] have tried to find the optimal threshold values to reduce the distortion. They use two threshold values where one is for positive and the other one for negative threshold values. Since their two threshold values are inappopriate to find optimum solution, our approach tries to use four threshold values. This paper shows that using four threshold values are more effective than two threshold values and that achieving the largest embedding capacity under the lowest distortion is possible as long as four threshold values are used under the two kinds of non-overlapping groups are exploited.As a result, this paper shows that our algorithm can produce better results than existing algorithms.

The paper is organized as follows. Section 2 introduces the proposed reversible data hiding technique which combines several methods such as prediction method, sorting method, and the optimum histogram pair shifting method. In section 3, we present experimental results to prove the superiority of the method compared to other methods. Conclusion is presented in Section 4.

2. Proposed Method

2.1 Diamond Prediction

The proposed method of this paper is based on a prediction and sorting method of Sachnev et al [6]. Their scheme is outstanding in terms of embedding capacity and distortion over latest technologies. Their approach has achieved the largest embedding capacity and smallest distortion so far. Thus, our objective is to find a solution producing better result than their method. We present the basis of their idea briefly in this subsection to make our method more comprehensible. In this paper, the capital character means a modified pixel value while the lowercase character means an original pixel value.

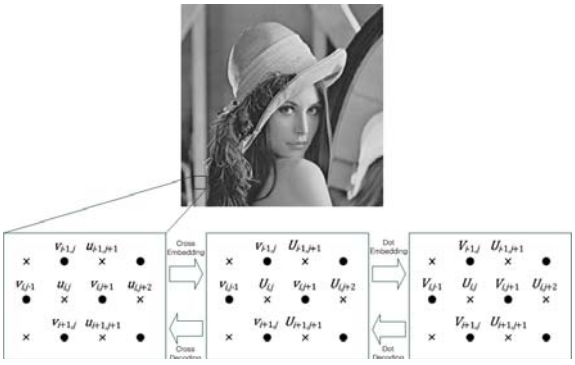

Fig. 1. Diamond prediction scheme.

Sachnev et al. [6] have proposed the diamond pattern prediction method with four pixels in each group. They divide the patterns into two groups: The dot set and the cross set. These two

sets are independent. In order to predict the pixel value of the position (i, j) in **Fig. 1**, four neighboring pixels $v_{i,j-1}, v_{i+1,j}, v_{i,j+1}$, and $v_{i-1,j}$ are used. The predicted value $u'_{i,j}$ in the position (i, j) is calculated as follows:

$$u'_{i,j} = \left\lfloor \frac{v_{i,j-1}+v_{i+1,j}+v_{i,j+1}+v_{i-1,j}}{4} \right\rfloor \tag{1}$$

When the original pixel value of $u'_{i,j}$ is $u_{i,j}$, its predicted error $d_{i,j}$ is obtained as follows:

$$d_{i,j} = u_{i,j} - u'_{i,j} \tag{2}$$

The predicted error can be expanded to embed a bit, b, according to the difference expansion algorithm [4] as follows:

$$D_{i,j} = 2d_{i,j+1} + b. \tag{3}$$

After data embedding, the original value, $u_{i,j}$ is modified to $U_{i,j}$ as follows:

$$U_{i,j} = D_{i,j} + u_{i,j}. \tag{4}$$

Similarly, $u_{i-1,j+1}$, $u_{i+1,j+1}$, and $u_{i,j+2}$ are modified to $U_{i-1,j+1}$, $U_{i+1,j+1}$ and $U_{i,j+2}$, respectively. This process is called 'cross embedding' in **Fig. 1**. After the cross embedding, we can obtain the modified pixel values $V_{i,j+1}$ from $v_{i,j+1}$ using the modified values of the cross set. Similarly, $v_{i-1,j}$, $v_{i+1,j}$, and $v_{i,j-1}$ are modified to $V_{i-1,j}$, $V_{i+1,j}$, and $V_{i,j-1}$, respectively, using Equations (1) to (4). This process is called 'dot embedding' in **Fig. 1**.

The decoding procedure is implemented from the dot set to the cross set in reverse order of embedding. In **Fig. 1**, decoding scheme for the dot set is an inverse of the embedding scheme. At first, the predicted value $v'_{i,j+1}$ is computed as follows using the neighboring four pixel values.

$$v'_{i,j+1} = \left\lfloor \frac{U_{i,j}+U_{i-1,j+1}+U_{i,j+2}+U_{i+1,j+1}}{4} \right\rfloor \tag{5}$$

The modified prediction error is computed as follows:

$$D_{i,j+1} = V_{i,j+1} - v'_{i,j+1} \tag{6}$$

The embedded bit can be extracted as follows:

$$b = D_{i,j+1} \bmod 2. \tag{7}$$

The original predicted error is computed as follows:

$$d_{i,j+1} = \left\lfloor \frac{D_{i,j+1}}{2} \right\rfloor \tag{8}$$

The original pixel value in the position $(i,j+1)$ is recovered as follows:

$$v_{i,j+1} = d_{i,j+1} + v'_{i,j+1} \tag{9}$$

Similarly, $V_{i-1,j}$, $V_{i+1,j}$, and $V_{i,j-1}$ are recovered to $v_{i-1,j}$, $v_{i+1,j}$, and $v_{i,j-1}$, respectively. This process is called 'dot decoding' in **Fig. 1**. After the dot decoding process, we can recover the pixel value $u_{i,j}$ from $U_{i,j}$. Similarly, $U_{i-1,j+1}$, $U_{i+1,j+1}$ and $U_{i,j+2}$ are recovered to $u_{i-1,j+1}$, $u_{i+1,j+1}$, and $u_{i,j+2}$, respectively. This process is called 'cross decoding' in **Fig. 1**.

2.2 Sorting by Local Variance

In the previous reversible data hiding algorithm of Sachnev et al. [6], the mean value of of the differences between four neighboring pixel values and the predicted value has been used for sorting. In this paper, we propose to sort the predicted errors according to the local variance value. For the cross set, the local variance value at the position (i, j) is calculated as follows:

$$\mu_{i,j} = \frac{1}{4}\sum_{k=1}^{4}\left(u'_{i,j} - v_k\right)^2 \tag{10}$$

where $u'_{i,j}$ is computed using Equation (1), and $v_1 = v_{i,j-1}$, $v_2 = v_{i+1,j}$, $v_3 = v_{i,j+1}$, and $v_4 = v_{i-1,j}$. Likewise, for the dot set, the local variance at the position $(i, j+1)$ is computed as follows:

$$\mu_{i,j+1} = \frac{1}{4}\sum_{k=1}^{4}\left(v'_{i,j+1} - U_k\right)^2 \tag{11}$$

where $v'_{i,j+1}$ is computed using Equation (5), $U_1 = U_{i,j}$, $U_2 = U_{i-1,j+1}$, $U_3 = U_{i,j+2}$, and $U_4 = U_{i+1,j+1}$.

The local variance value μ has two important features. First, this value remains unchanged after data embedding. Thus, the local variance value can be used for restoring the pixels to the original status. Second, this value is, in general, proportional to the magnitude of the predicted error $d_{i,j}$. Therefore, the group of pixels with small variance value generally has a small magnitude of predicted errors. After sorting, the groups of pixels are arranged in ascending order according to the local variance values. As a result, total distortion gets smaller since many smaller predicted errors can be used for embedding data.

2.3 Histogram Pair Shifting

Traditional histogram shifting (HS) method uses T_n and T_p to control the embedding (i.e., expanding) regions and the shifting region. On the other hand, in this paper, the histogram pair shifting (HPS) method uses T_{n2}, T_{n1}, T_{p1}, and T_{p2} for its embedding algorithm. The threshold values meet the following condition: $T_{n2} \le T_{n1} \le T_{p1} \le T_{p2}$. The HPS embedding algorithm modifies the predicted error as follows:

$$D=\begin{cases} d+N+1, & if \quad d<T_{n2} \\ 2(d-T_{n1})+T_{n1}-1+b, & if \quad T_{n2}\le d\le T_{n1} \\ d, & if \quad T_{n1}<d<T_{p1} \\ 2(d-T_{p1})+T_{p1}+b, & if \quad T_{p1}\le d\le T_{p2} \\ d+P+1, & if \quad d<T_{p2} \end{cases} \tag{12}$$

where $N = T_{n2} - T_{n1}$, $P = T_{p2} - T_{p1}$, and, b is a bit to embed. The decoder recovers the original predicted error and extracts the embedded data b from d as follows:

$$d=\begin{cases} D-N+1, & if \quad D<T_{n1}+2N-1 \\ D-\lfloor (D-T_{n1})/2 \rfloor, & if \quad T_{n2}+2N-1\le D\le T_{n1} \\ D, & if \quad T_{n1}<D<T_{p1} \\ D-\lfloor (D-T_{p1})/2 \rfloor, & if \quad T_{p1}\le D\le T_{p1}+2P+1 \\ d-P-1, & if \quad T_{p1}+2P+1<D \end{cases} \tag{13}$$

and

$$b=\begin{cases} D-2(d-T_{n1})-T_{n1}+1, & if \quad T_{n2}+2N-1\le D\le T_{n1} \\ D-2(d-T_{p1})-T_{p1}, & if \quad T_{p1}\le D\le T_{p1}+2P+1 \end{cases} \tag{14}$$

An example is provided to facilitate understanding of this algorithm. Assume that T_{n2} = -2, T_{n1} = -1, and $T_{p2} = T_{p1}$ = 2. In this case, the original predicted error d is modified to D as shown in **Fig. 2**. In this figure, note that the predicted errors 0 and 1 are unchanged since they are in between T_{n1} and T_{p1}. The value 2 can be modified to either 2 or 3 according to the hidden message. Similarly, the values -1 and -2 can be expanded to -1 or -2 and -3 or -4, respectively. The values larger than T_{p2} are shifted. **sdfasdf** $\boldsymbol{D_{i,j}}$ $\boldsymbol{d_{i,j}}D < T_{n1} + 2N - 1d = D - N + 1T_{n1} + 2N - 1 \le D \le T_{n1}d = D - \lfloor (D - T_{n1})/2 \rfloor b = D - 2(d - T_{n1}) - T_{n1} + 1T_{n1} < D < T_{p1}d = DT_{p1} \le D \le T_{p1} + 2P + 1d = D - \lfloor (D - T_{p1})/2 \rfloor b = D - 2(d - T_{p1}) - T_{p1}T_{p1} + 2P + 1 < DD = d - P - 1$

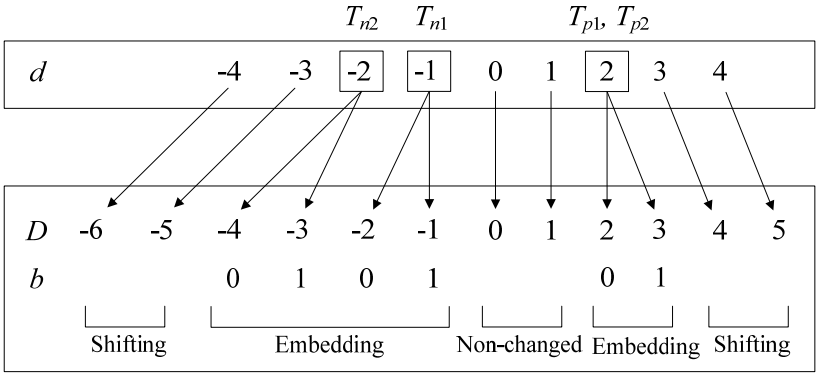

Fig. 2. A simple example of HPS embedding algorithm.

2.4 Example for Optimum Histogram Pair Shifting

Using a simple example, we can show the reason why the optimum histogram pair shifting has less distortion than the normal histogram shifting. In **Fig. 3**, we present a simple example for comparing the HS method with HPS method. Most images have a histogram of *d* values with Laplace distribution. In the first (i.e., upper) histogram, *x*-axis shows the predicted error value *d*, and *y*-axis the number of occurrences of *d*. In the second and third (i.e., bottom) histograms, the *x*-axis shows the modified predicted error value *D*, and the *y*-axis the number of occurrences of *D*. To compare fidelity with the original image, the peak signal-to-noise ratio (PSNR) is generally used. In this example, we show mean squared error (MSE) to calculate the distortion of each method. (Proposed algorithm also utilizes MSE to compute the distortion.)

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left\|I_{org(i,j)} - I_{mod(i,j)}\right\|^2 \tag{15}$$

where, I_{org} is an $m \times n$ original image, and I_{mod} is an $m \times n$ modified image. Note that $I_{org(i,j)} - I_{mod(i,j)} = (d_{i,j} + u'_{i,j}) - (D_{i,j} - u'_{i,j}) = d_{i,j} - D_{i,j}$.

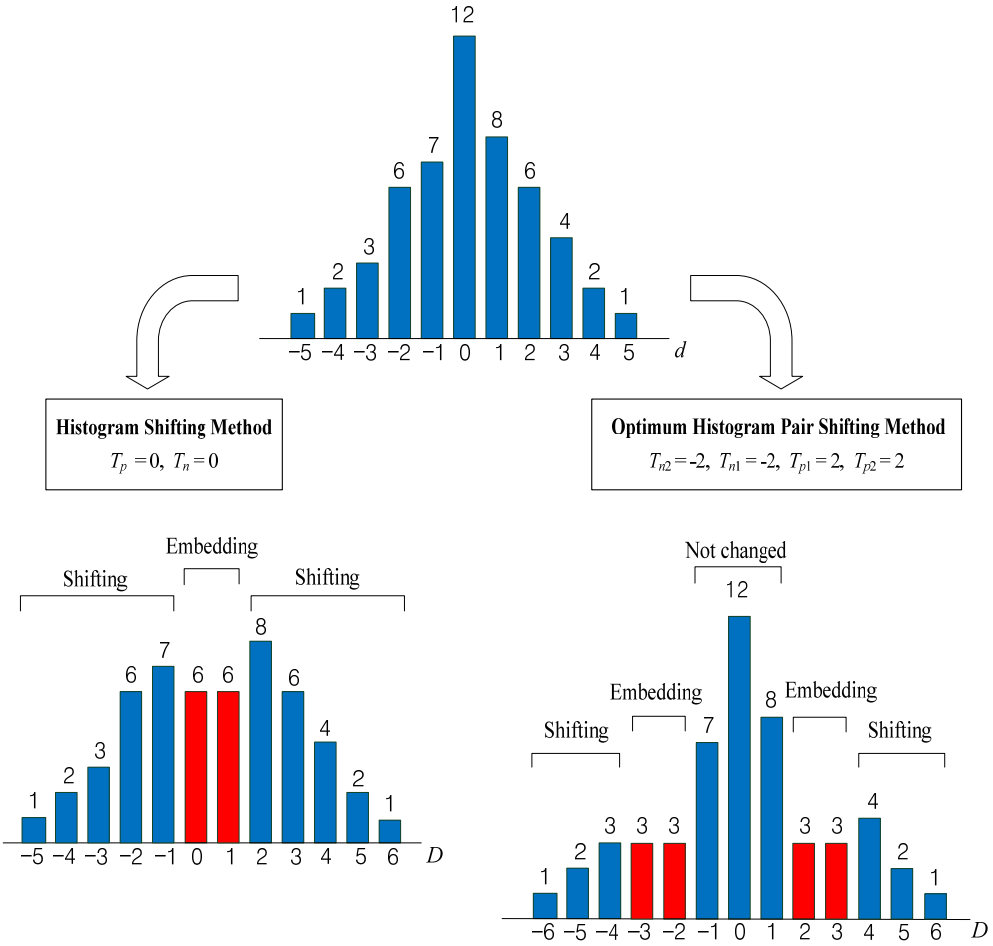

Fig. 3. Comparison between the HS and HPS methods.

In **Fig. 3**, in case of the HS method, only positive part of the histogram is modified. Its mean squared error (MSE_{HS}) is calculated as follows: ($\frac{1}{mn}$ is omitted.)

$$\mathrm{MSE}_{HS} = 6 \times 0^2 + 6 \times 1^2 + 8 \times 1^2 + 6 \times 1^2 + 4 \times 1^2 + 2 \times 1^2 + 1 \times 1^2 = 27.$$

In this case, one peak at $d = 0$ is used to hide 12 bits. All bins in the right of the peak are moved to the right by one position. Therefore, we can hide 12 bits into two bins with $D = 0$ and $D = 1$. We assume that there are six 1's and six 0's to hide. Six bits are hidden into the bin with $D = 0$. In this case, there is no distortion error. Six bits are hidden into the bin with $D = 1$. Thus, all the positive bins cause the distortion errors of magnitude 1 for each value.

In **Fig. 3**, the HPS method empties the histogram at two positions. In this case, the positions at $d = 3$ and $d = -3$ are emptied. The positions should be selected to minimize the total distortion. Non-changed region that has no distorton is formed in $-1 \leq D \leq 1$. Thus, there is no distortion of the values in the three bins.

The accumulated mean squared error in the positive side and negative side is 19 (i.e., 10 and 9, respectively). Compared with the HS method, the squared error of the HPS is smaller under the same embedding capacity.

$$\mathrm{MSE}_{positive} = 3 \times 0^2 + 3 \times 1^2 + 4 \times 1^2 + 2 \times 1^2 + 1 \times 1^2 = 10.$$
$$\mathrm{MSE}_{negative} = 3 \times 0^2 + 3 \times 1^2 + 3 \times 1^2 + 2 \times 1^2 + 1 \times 1^2 = 9.$$
$$\mathrm{MSE}_{HPS} = \mathrm{MSE}_{positive} + \mathrm{MSE}_{negative} = 19.$$

2.5 Finding Optimum Thresholds

In **Fig. 3**, the optimum threshold values of the HPS method is given as follows: $T_{n2} = T_{n1} = -2$, and $T_{p1} = T_{p2} = 2$. The optimum threshold values depend on the payload size and the hidden message as well. In addition, not all d values can be shifted or expanded due to the overflow/underflow problems. Thus, finding the optimal solution is a challenging problem.

For a given payload size, there can be many possible combinations of threshold values. In order to achieve as high PSNR value as possible, it is necessary to find the optimum threshold values. However, we have to consider embedding capacity, composition of secret message, overflow/underflow problems, and the cross and dot sets altogether, which is too challenging to solve. Therefore, we present an algorithm for finding the optimum threshold values in this paper. This algorithm finds the optimum threshold values for both positive and negative sides. In addition, it should also work properly both for the cross and dot sets. In other words, it produces the optimum threshold values for each set. It is a kind of brute-force search approach by adjusting the threshold values iteratively.

The algorithm for finding the optimum threshold values of the cross set is shown in **Fig. 4**. After dividing the image into the cross and dot sets, our algorithm starts from the cross set. Sorting d values in ascending order allows us to utilize the small d values first. (Small d causes small distortion.) The initial threshold values are $T_{n1} = 0$, $T_{n2} = 0$, $T_{p1} = 1$, and $T_{p2} = 1$. Under this condition, this algorithm checks if embedding capacity to hide is sufficient. If these threshold values are sufficient to hide a required number of bits, its MSE value is computed and recorded. This algorithm tries to choose appropriate threshold values based on the lowest MSE value. Thus, this algorithm updates the threshold values when,

1. the threshold values are too small to hide a required number of bits, or
2. the current MSE value is larger than the lowest MSE value obtained before.

The adjustment rule of threshold values is simple. In this case, at first, we fix the three values and increase the value T_{p2} by 1 in each iterative step such as 1, 2, 3, and so on. Next, another threshold value T_{p1} is increased by 1 as $T_{p1} = 2$ and the value T_{p2} by 1 in each iterative step such as 2, 3, 4, and so on. In other words, this algorithm finds the optimum threshold

values iteratively using brute-force search method. However, the number of iterations can be reduced using an intelligent inference. The MSE values have an interesting tendency that they decrease up to a certain value and then increase drawing U curve. Thus, the variable threshold value should not be increased after the certain value. The same rule is applied to the negative side threshold values T_{n1} and T_{n2}. In some cases, increasing the threshold values accordingly increases the accumulated errors. It implies that further search is not necessary for optimization.

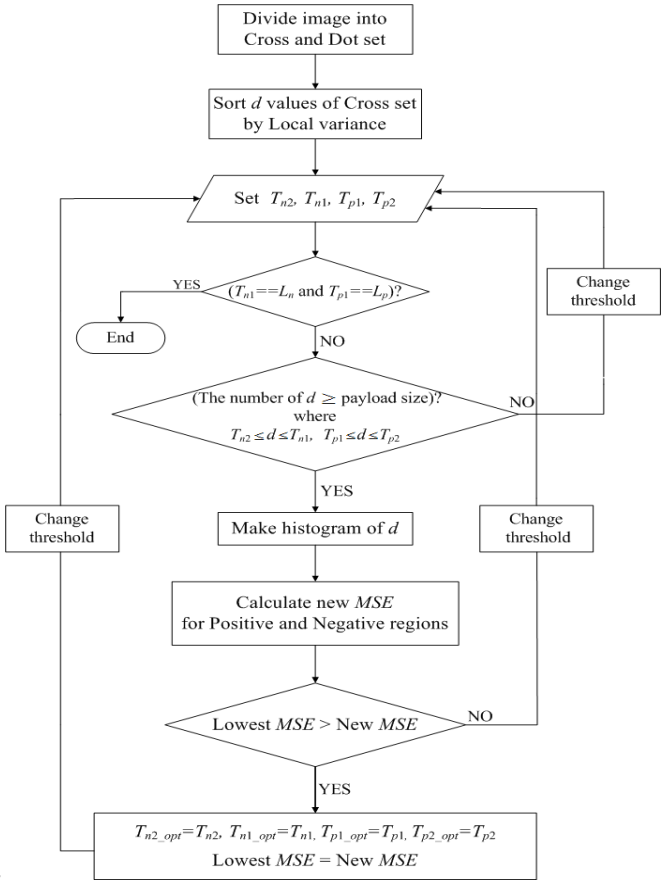

Fig. 4. The algorithm for finding optimum thresholds of cross set.

Table 1 shows the optimum threshold values which are obtained by our algorithm for three typical images. Note that the threshold values are different from images and either the cross set or dot set.

Table 1. Optimum threshold values

Payload (bits)		Lena	Mandrill	Airplane
		$T_{n2}, T_{n1}, T_{p1}, T_{p2}$	$T_{n2}, T_{n1}, T_{p1}, T_{p2}$	$T_{n2}, T_{n1}, T_{p1}, T_{p2}$
20k	Cross set	-3, -3, 3, 3	-4, -3, 2, 2	0, 0, 1, 1
	Dot set	-2, -2, 3, 3	-2, -2, 3, 4	-1, -1, 2, 2
40k	Cross set	-1, -1, 2, 2	-3, -1, 2, 4	0, 0, 1, 1
	Dot set	-2, -2, 2, 2	-3, -1, 3, 5	-1, -1, 2, 2

60k	Cross set	-1, -1, 1, 1	-3, 0, 1, 4	0, 0, 1, 1
	Dot set	-1, -1, 1, 1	-4, -1, 2, 5	0, 0, 2, 2
80k	Cross set	-2, -1, 1, 1	-5, 0, 1, 6	0, 0, 1, 1
	Dot set	-2, -1, 2, 3	-4, 0, 2, 7	0, 0, 1, 1
100k	Cross set	-2, -1, 1, 2	-7, 0, 1, 8	0, 0, 1, 2
	Dot set	-2, -1, 1, 2	-6, 0, 1, 7	0, 0, 1, 2
240k	Cross set	-9, 0, 1, 10	-31, 0, 1, 32	-9, 0, 1, 10
	Dot set	-9, 0, 1, 10	-34, 0, 1, 35	-9, 0, 1, 10

2.6. Overflow and Underflow Problems

The overflow/underflow problem has been one of the biggest issues of the reversible data hiding algorithms. To meet the reversibility requirement, we have to use the location map which is a kind of side information. The additional information (such as the location map) should be as small as possible for large embedding capacity and less distortion. The location map should be included in the payload. Thus, the gross payload includes the net payload plus the location map.

In the proposed method, the overflow/underflow problem results from the following condition:

$$D_{i,j} + u'_{i,j} > 255 \text{ or } D_{i,j} + u'_{i,j} < 0 \tag{16}$$

In this case, data embedding is not possible into the pixel at the position (i, j). The pixel value should remain unchanged. The location map records the position for detectors not to try to extract hidden messages. After data hiding, at most of the positions, decoder can extract the hidden message bits and recover the modified values without ambiguity. However, in a few positions, the decoder has to rely on the location map to tell whether the values have been modified or unmodified when the predicted errors are on the suspicious border. The border depends on the threshold values and images.

In case of **Fig. 2**, when $d_{i,j} = 2$, $D_{i,j} = 3$ (the embedded bit should be '1' to test overflow problem with the hardest case), $u_{i,j} = 255$ and $u'_{i,j} = 253$, then, the overflow problem takes place as follows: $I_{mod(i,j)} = D_{i,j} + u'_{i,j} = 257$. Therefore, this pixel value should be unchanged such that $I_{mod(i,j)} = I_{org(i,j)} = 255$. The location map is needed to resolve this kind of problem. On the other hand, if $D_{i,j+1} = 3$ and $u'_{i,j+1} = 252$, then $I_{mod(i,j+1)} = 255$. The location map is also needed to distinguish two different cases: $I_{mod(i,j)}$ from $I_{mod(i,j+1)}$. The location map informs the decoder that the first case has not been modified and the second case modified.

The algorithm to make the location map in the encoder is as follows:

(1) At the position (i, j), modify $d_{i,j}$ to $D_{i,j}$ using Equation (12). Obtain $I_{mod(i,j)} = D_{i,j} + u'_{i,j}$.

(2) If $I_{mod(i,j)}$ does not cause overflow/underflow problem according to Equation (16),

  (a) Modify $D_{i,j}$ to D_{test} using Equation (12). Obtain $I_{test(i,j)} = D_{test(i,j)} + u'_{i,j}$. For making the hardest case, embed a bit '1' in the positive embedding region and a bit '0' in the negative embedding region.

  (b) If $I_{test(i,j)}$ does not cause overflow/underflow problem, it means that no bit is

needed to mark into the location map for this pixel, and $I_{mod(i,j)} = D_{i,j} + u'_{i,j}$. Then, go to step 1 to process the next pixel.

(c) If $I_{test(i,j)}$ causes overflow/underflow problem, add a bit '0' in the location map and $I_{mod(i,j)} = D_{i,j} + u'_{i,j}$. Then, go to step 1 to process the next pixel.

(3) If $I_{mod(i,j)}$ causes overflow/underflow problem according to Equation (16), add a bit '1' in the location map and $I_{mod(i,j)} = I_{org(i,j)} = d_{i,j} + u'_{i,j}$. This is the case that the original value of the pixel is not modified at all. Then, go to step 1 to process the next pixel.

The algorithm to use the location map in the decoder is as follows:

(1) At the position (i, j), modify $D_{i,j}$ to D_{test} using Equation (12). Obtain $I_{test(i,j)} = D_{test(i,j)} + u'_{i,j}$. from $I_{mod(i,j)} = D_{i,j} + u'_{i,j}$.

(2) If $I_{test(i,j)}$ does not cause overflow/underflow problem, it means that location map is not necessary to decode in this case. Then, go to step 1 to process the next pixel.

(3) If $I_{test(i,j)}$ causes overflow/underflow problem, check the present bit of the location map.

(a) If the current bit in the location map equals to '1', the recovered pixel value is as follows: $I_{rec(i,j)} = I_{mod(i,j)}$. It means that this pixel has been unchanged at the encoder. Then, go to step 1 to process the next pixel.

(b) If the current bit in the location map equals to '0', modify $D_{i,j}$ to $d_{i,j}$ using Equation (13). Obtain $I_{rec(i,j)} = d_{i,j} + u'_{i,j}$ from $I_{mod(i,j)} = D_{i,j} + u'_{i,j}$. Then, go to step 1 to process the next pixel.

Table 2 shows the location map sizes for 3 typical test images. Note that the Airplane image does not request the location map due to its unique histogram characteristics. This image does not have extreme pixel values such as around 0 or 255. Lena image is also similar to the Airplane image. However, the Mandrill image is different. This image has many extreme values at both ends (i.e., 0 and 255). Thus, this image requeusts relatively large number of location map as the embedding capacity increases.

Table 2. Location map size

Payload (bits)	Location map(bits)		
	Lena	Mandrill	Airplane
40k	0	1	0
80k	0	17	0
120k	0	125	0
160k	0	278	0
200k	2	1,149	0
240k	4	9,482	0

2. 7. Encoder and Decoder

This section presents the encoding and decoding processes. A simple block diagram shown in **Fig. 7** summarizes all processes of the proposed method. Note that embedded message in I_{mod} includes the side information indicating the threshold values, gross payload size of each set (i.e., the cross and dot sets, respectively), and location map size, as well as P_{cross}, P_{dot}, and

location map bits. The side information is embedded in according to LSB substitution [9], in the beginning part of message. Each threshold value (T_{n2}, T_{n1}, T_{p1}, and T_{p2}) needs 6 bits which is able to represent up to 63($= 2^6 - 1$) and sufficient for expressing expected threshold values. Note that **Table 1** show that the maximum threshold value in magnitude is 35 for the Mandrill image to hide 240,000 bits (0.92 bpp). The location map size can be represented by 10 bits. In addition, 17 bits are allocated to designate cross and dot payload size, respectively, because 18 bits must be sufficient to indicate the typical size of images (512×512, grayscale). In conclusion, the side information for each set consists of four threshold values in magnitude (24 bits), location map size (10 bits), and payload size (17 bits). The original 51 LSB values should be included to the payload, before embedding side information.

In decoding process, the side information is extracted from the LSB values of the first 51 sorted prediction errors of each set. From the 52nd sorted prediction error, payloads, location map bits, and the original 51 LSB values, are extracted in according to the equation (14).

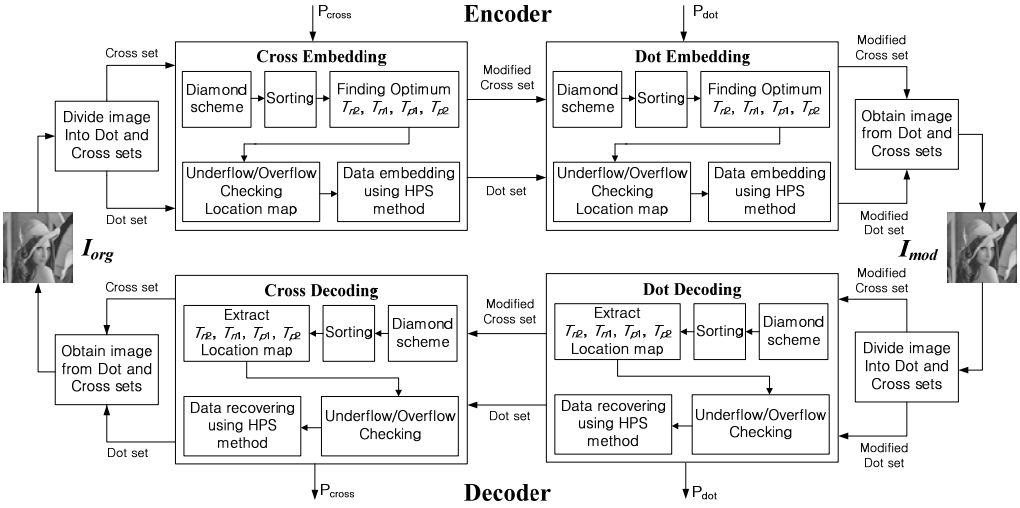

Fig. 7. Encoder and decoder for proposed method.

3. Experimental Result

The propoesd method is compared with the works of Sachnev et al. [6], Xuan et al. [7], Thodi and Rodriguez [4], Lee et al. [9], Kamstra and Heijmans [3] using typical 512×512 grayscale images such as Lena, Mandrill and Airplane. The comparison results are shown in **Fig. 8**, **Fig. 9**, and **Fig. 10** for each image. Left columns in these figures show the results in terms of embedding capacity up to 0.5 bpp while right columns small capacity up to 0.03 bpp. In these figures, it is obvious that our approach is better than exisiting methods. The gap between ours and Sachnev et al.'s method [6] are very close. However, ours are mostly better than theirs.

In particular, the proposed method shows much better performance in small embedding capacity. As illustrated in **Fig. 3,** when the modification of histogram is small, finding the optimum threshold values works more easily since the search space is narrower. As we expected, improvements over other methods are more significant in the right column in the graphs which deal with small capacities. The exact experimental data are summarized in **Table 3** and **Table 4**.

Performance comparison with Hong et al.`s method [5] is shown in **Table 5**, **Table 6**, and **Table 7**. In most cases, our method outperforms theirs in terms of PSNR value. The proposed

method is far better than their method. One exception is on Lena image where embedding capacity is 86,178 bits.

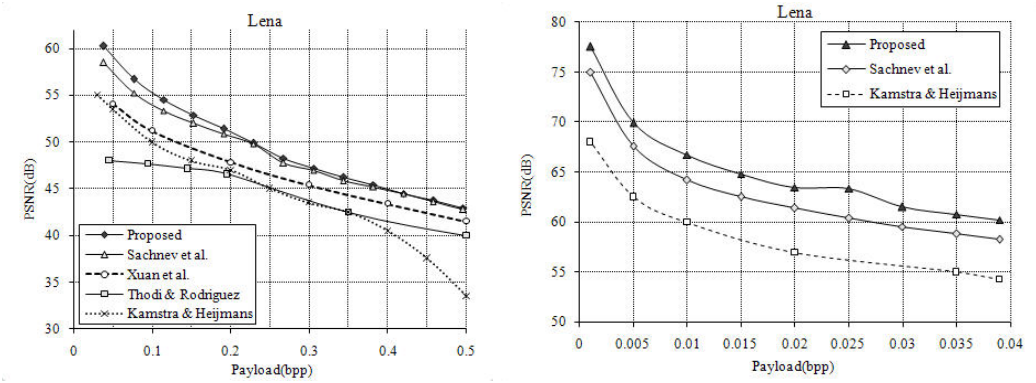

Fig. 8. Performance comparison on lena.

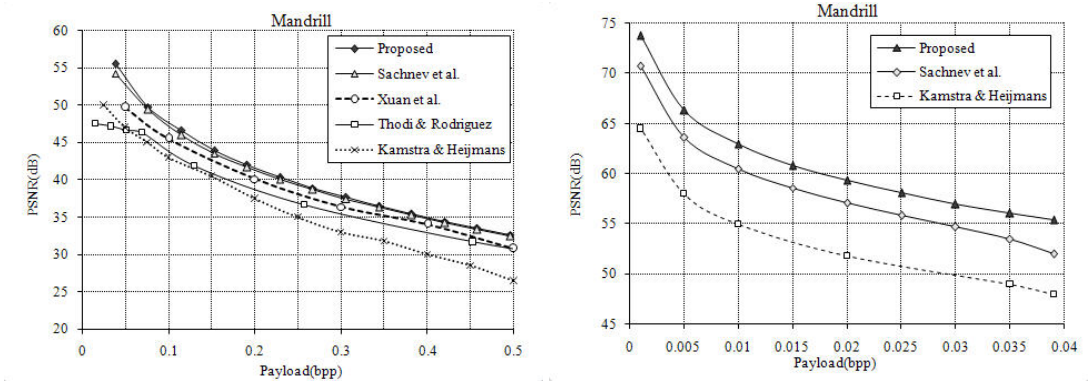

Fig. 9. Performance comparison on mandrill.

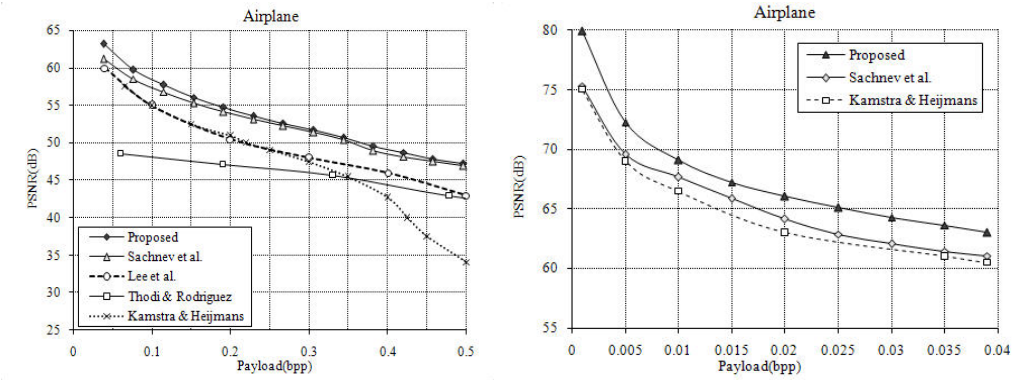

Fig. 10. Performance comparison on airplane.

Table 3. PSNR values of the proposed method

Payload		**PSNR(dB)**					
bpp	bits	Lena		Mandrill		Airplane	
		Proposed	Sachnev et al. [6]	Proposed	Sachnev et al. [6]	Proposed	Sachnev et al. [6]
0.04	10k	60.34	58.51	55.54	54.23	63.19	61.24
0.08	20k	56.78	55.29	49.80	49.45	59.80	58.55
0.11	30k	54.58	53.36	46.60	46.04	57.72	56.80
0.15	40k	52.89	52.01	43.95	43.53	56.09	55.30
0.19	50k	51.42	50.88	41.98	41.73	54.68	54.15
0.23	60k	49.88	49.78	40.36	40.06	53.53	53.14
0.27	70k	48.31	47.74	38.92	38.66	52.59	52.24
0.31	80k	47.21	46.96	37.66	37.44	51.71	51.40
0.34	90k	46.20	45.83	36.52	36.31	50.73	50.37
0.38	100k	45.45	45.21	35.47	35.25	49.51	48.95
0.42	110k	44.50	44.47	34.48	34.27	48.74	48.17
0.46	120k	43.74	43.59	33.54	33.35	47.80	47.48
0.50	130k	42.96	42.78	32.67	32.46	47.19	46.91
0.53	140k	42.31	42.29	31.78	31.62	46.52	46.22

Table 4. PSNR values of the proposed method with small embedding capacities

Payload		**PSNR(dB)**					
bpp	bits	Lena		Mandrill		Airplane	
		Proposed	Sachnev et al. [6]	Proposed	Sachnev et al. [6]	Proposed	Sachnev et al. [6]
0.00	264	77.5450	75.00	73.7975	70.78	79.9107	75.31
0.00	1,312	69.9207	67.59	66.3032	63.66	72.2859	69.57
0.01	2,622	66.7604	64.25	62.9255	60.46	69.0713	67.65
0.01	3,934	64.8320	62.52	60.8050	58.60	67.2403	65.88
0.02	5,244	63.4870	61.42	59.3029	57.11	66.0878	64.13
0.02	6,554	63.3660	60.41	58.0859	55.86	65.1362	62.88
0.03	7,886	61.5443	59.58	56.9787	54.67	64.3138	62.06

Table 5. Performance comparison on lena

Payload (bits)	**PSNR(dB)**	
	Proposed	Hong et al. [5]
6,657	62.30	61.80
86,178	46.50	48.93

Table 6. Performance comparison on mandrill

Payload (bits)	**PSNR(dB)**	
	Proposed	Hong et al. [5]
5,685	58.86	51.79
16,575	51.86	48.29

Table 7. Performance comparison on airplane

Payload (bits)	PSNR(dB)	
	Proposed	Hong et al. [5]
16,974	60.61	55.14
71,610	52.45	48.79

4. Conclusions

In this paper, an improved reversible data hiding technique is presented. This approach is based on the non-overlapping cross and dot sets of diamond predictors, sorting, expanding, and shifting methods of the Sachnev et al.'s approach [6]. We utilize four threshold values to control the mean-squared error under the given embedding capacity. We show that our approach can produce better results than Sachnev et al.'s approach [6] which is the best one so far. It is obvious that optimal threshold values can produce the optimal results under the condition that four threshold values are used over the errors computed based on the non-overlapping cross and dot sets of diamond predictors. Experiment results show that ours are the best in terms of embedding capacity and the distortion (i.e., less distorted under the given embedding capacity).

References

[1] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890-896, 2003.
[2] H. J. Kim, V. Sachnev, Y. Q. Shi, J. Nam, and H. G. Choo, "A novel difference expansion transform for reversible data hiding," *IEEE Transactions on Information Forensics and Security*, vol 3, no. 3, pp. 456-465, 2008.
[3] L. H. J. Kamstra, and A. M. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2082-2090, 2005.
[4] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 721-730, 2007.
[5] W. Hong, T. S. Chen, and C. W. Shiu, "Reversible data hiding for high quality images using modification of prediction errors," *The Journal of Systems and Software*, vol. 82, pp. 1833-1842, 2009.
[6] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y.-Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Transaction on Circuit System and Video Technology*, vol. 19, no. 7, pp. 989-999, 2009.
[7] G. Xuan, Yun Q. Shi, P. Chai, X. Cui, Z. Ni, and X. Tong, "Optimum histogram pair based image lossless data embedding," *Lecture Notes in Computer Science*, vol. 5041, pp. 264-278, 2008.
[8] Ran-Zan Wang, Chi-Fang Linb, and Ja-Chen Lina"Image hiding by optimal LSB substitution and genetic algorithm," *Pattern Recognition*, vol. 34, no. 3, pp. 671-683., 2001.
[9] S. Lee, C. D. Yoo, and T. Kalker, "Reversible image watermarking based on integer-to-integer wavelet transform," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 321-330, 2007.
[10] A. M. Alattar, "Reversible watermark using difference expansion of triplets," in *Proceedings of the International Conf. on Image Processing*, vol. 1, pp. 501-504, 2003. W. Hong, T. S. Chen, and C. W. Shiu, "Reversible data hiding for high quality images using modification of prediction errors," *The Journal of Systems and Software*, vol. 82, pp. 1833-1842, 2009.

Hee Joon Hwang received a B.S. degree in Electric and Electronic Engineering department and a M.S. degree in Graduate School of Information Management and Security from Korea University, Seoul, Korea, in 2008. He joined Multimedia Security Laboratory at the Center of Information Security and Technology (CIST), Graduate School of Information Management and Security, Korea University, Seoul, Korea at 2007, where he is currently pursuing Ph.D. His research interests include multimedia security, reversible and robust watermarking, steganography.

Hyoung Joong Kim received his B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, Korea, in 1978, 1986 and 1989, respectively. He joined the faculty of the Department of Control and Instrumentation Engineering, Kangwon National University, Korea, in 1989. He is currently a Professor of the Graduate School of Information Management and Security, Korea University, Korea since 2006. His research interests include parallel and distributed computing, multimedia computing, and multimedia security.

Vasiliy Sachnev received his B.S and M.S. degrees in Electrical Engineering from the Komsomolsk-na-Amure State Technical University, Russia, in 2002 and 2004. He joined Multimedia Security Laboratory at the Center of Information Security and Technology (CIST), Graduate School of Information Management and Security, Korea University, Seoul, Korea at 2007, where he received Ph. D degree in 2009 and he is currently pursuing Post Doc. His research interests include multimedia security, digital watermarking, steganography, and image processing.

Sang Hyun Joo received the B.S and MS from Dongguk University, Seoul, Korea in 1989 and 1994, and Ph.D, from Niigata University, Niigata, Japan, in electrical engineering in 1999, respectively. From 1994 to 1996, he worked as a research engineer in KaiTech. From 1999 to 2001, he worked as a research associate in Niigata University. After that, he joined ETRl and is currently a principal engineering staff in contents division. Since 2007, he has worked for MPEG RoSE and MPEG-V(ISO/IEC 23005) as an ad hoc group chairman and an editor. His research Interests have been watermarking, fingerprints, media context and control, communication technologies between real and virtual worlds including AR, VR, and MR.