

A High-Capacity, Invertible, Data-Hiding Algorithm Using a Generalized, Reversible, Integer Transform

John Stach and Adnan M. Alattar
Digimarc Corporation, Tualatin, OR 97062

ABSTRACT

A high-capacity, data-hiding algorithm that lets the user restore the original host image after retrieving the hidden data is presented in this paper. The proposed algorithm can be used for watermarking valuable or sensitive images such as original art works or military and medical images. The proposed algorithm is based on a generalized, reversible, integer transform, which calculates the average and pair-wise differences between the elements of a vector extracted from the pixels of the image. The watermark is embedded into a set of carefully selected coefficients by replacing the least significant bit (LSB) of every selected coefficient by a watermark bit. Most of these coefficients are shifted left by one bit before replacing their LSBs. Several conditions are derived and used in selecting the appropriate coefficients to ensure that they remain identifiable after embedding. In addition, the selection of coefficients ensures that the embedding process does not cause any overflow or underflow when the inverse of the transform is computed. To ensure reversibility, the locations of the shifted coefficients and the original LSBs are embedded in the selected coefficients before embedding the desired payload. Simulation results of the algorithm and its performance are also presented and discussed in the paper.

Keywords: Difference Expansion, Data Embedding, Invertible Watermarks, Reversible Watermark, Steganography

1. INTRODUCTION

Watermarking valuable and sensitive images such as original novel artworks and military and medical images represents a challenge to most watermarking algorithms. First, these applications require the embedding of several kilobytes of data, while most robust watermarking algorithms can only embed several hundreds of bits. Second, the watermarking process usually introduces a slight but irreversible degradation in the original image. This degradation may reduce the aesthetic and monetary values of the artwork. It may also cause the loss of significant artifacts in military and medical images. These artifacts may be crucial for an accurate analysis of the military images or for an accurate diagnosis of the medical images. As well, the degradation may introduce new misleading artifacts. The aforementioned applications, however, may be willing to tolerate the addition of some noise provided that the watermarking algorithm can embed the desired amount of data, and provided that the noise can be easily removed to restore the original host image without reference to any information beyond the watermarked image itself.

Several algorithms [1]-[11] have been proposed in the literature to provide solutions to this problem. Mintzer, et al. [1], used completely visible but removable patterns (low SNR) with their digital library application to promote image sale on the Internet. The user can download a free marked image as a “teaser”, but he or she must purchase a “vaccine” program to restore a high quality image from the teaser image. Honsinger, et al. [2], included the embedding parameters with a robust, additive, and non-adaptive watermark to devise a low-capacity reversible watermarking algorithm. These parameters are used by the watermark reader to calculate then subtract the watermark from the image. Macq [3] extended Honsinger’s technique to the Patchwork algorithm proposed by Bender, et al.[4], for a robust watermark. Both of Macq and Honsinger’s algorithms suffer from a wraparound effect that causes a salt-and-pepper visual artifact. Vleeschouwer, et al. [5], reduce the salt-and-pepper artifact in Macq’s method by using a circular interpretation of bijective transformations of the histogram of the blocks used for the Patchwork. The capacity of this algorithm depends on the size of the block used for the Patchwork, and it is less than 10 kB/image.

Fridrich, et al. [6], also devised a reversible watermark that does not suffer from the salt-and-pepper artifact. That algorithm compresses one of the least significant bit planes of the host image, appends an image hash and payload, encrypts the result, and finally, replaces the original bit plane with the result. Fridrich also extended the technique to

JPEG-compressed images [7], GIF and PNG palette images [8], and introduced a more efficient (than [6]) RS-embedding method for bitmap images. The capacity of the algorithms of Fridrich is low; it is several hundred bits/image.

Celik, et al. [9], enhanced Fridrich's approach and devised a low distortion, reversible watermark that is capable of embedding as high as 0.7 bits/pixel. His algorithm quantizes each pixel by a quantizer of L step size, compresses the quantization noise and appends a payload to it, and then adds an L -ary representation of the result to the quantized image.

Tian [10] used a difference expansion transform of a pair of pixels to devise a high-capacity, low-distortion, reversible watermark. His algorithm divides the image into pairs of pixels then embeds one bit into the difference of the pixels of each pair from those pairs that are not expected to cause an overflow or underflow. The location map that indicates the modified pairs is compressed and included in the payload. Alattar [11] extended Tian's algorithm to spatial and cross color triplets to reduce the size of the location map and to hide two bits in every three spatial or cross-color components pixels.

Kalker, et al. [12], derived capacity bounds for a reversible watermark and proposed using error correction codes to make the reversible watermark robust to ordinary processing [13].

In this paper, we generalize the algorithm we proposed in [11] to vectors of length more than 3 to further increase the capacity and the computation efficiency of the algorithm. The proposed algorithm is based on a generalized, reversible, integer transform (GRIT), which will be derived in the next section. In Section 3, we describe the proposed embedding and recovery algorithms for a reversible watermark. In Section 4, we present simulation results of the proposed algorithm using quad vectors. And finally, in the last section, we summarize our conclusions.

2. THE GENERALIZED, REVERSIBLE, INTEGER TRANSFORM

2.1. Theorem

If \mathbf{D} is an $N \times N$ full-rank matrix with an inverse \mathbf{D}^{-1} , \mathbf{u} is an $N \times 1$ integer column vector, and $\mathbf{D}\mathbf{u} = [a \ d_1 \ d_2 \ \cdots \ d_{N-1}]^T$, where a is the weighted average value of the elements of \mathbf{u} , and d_1, d_2, \dots, d_{N-1} are independent pair-wise differences between the elements of \mathbf{u} . Then $\mathbf{v} = \lfloor \mathbf{D}\mathbf{u} \rfloor$ and $\mathbf{u} = \left\lceil \mathbf{D}^{-1} \lfloor \mathbf{v} \rfloor \right\rceil$ form a GRIT pair, where $\lceil \cdot \rceil$ indicates the ceiling function (i.e., round up to nearest integer) and $\lfloor \cdot \rfloor$ indicates the floor function (i.e., round down to the nearest integer).

2.2. Proof

To satisfy $\mathbf{D}\mathbf{D}^{-1} = \mathbf{I}$, each element of the first column of \mathbf{D}^{-1} must be 1. This is the result because the inner product of the first row of \mathbf{D} and the first column of \mathbf{D}^{-1} must be 1, and the inner product of each of the remaining $N-1$ difference rows of \mathbf{D} and the first column of \mathbf{D}^{-1} must generate a 0. Therefore, \mathbf{D}^{-1} can be written as

$$\mathbf{D}^{-1} = \begin{bmatrix} 1 & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,N-2} & \alpha_{0,N-1} \\ 1 & \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,2} & \alpha_{1,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{N-2,1} & \alpha_{N-2,2} & \cdots & \alpha_{N-2,N-2} & \alpha_{N-1,N-1} \\ 1 & \alpha_{N-1,1} & \alpha_{N-1,2} & \cdots & \alpha_{N-1,N-2} & \alpha_{N-1,N-1} \end{bmatrix}$$

Since d_i 's are integers, $\mathbf{v} = \lfloor \mathbf{D}\mathbf{u} \rfloor = \lfloor \lfloor a \rfloor \quad d_1 \quad d_2 \quad \dots \quad d_{N-1} \rfloor^T$, and since $\mathbf{u} = \mathbf{D}^{-1}\mathbf{D}\mathbf{u}$, each element of u can be expressed explicitly as $u_i = a + \sum_{j=1}^{N-1} \alpha_{j,i} d_j$. Hence, $\sum_{j=1}^{N-1} \alpha_{j,i} d_j = u_i - a$. Now, an approximation of u can be

written as $\hat{\mathbf{u}} = \mathbf{D}^{-1}\mathbf{v}$. Each element of this approximation can be expressed explicitly as $\hat{u}_i = \lfloor a \rfloor + \sum_{j=1}^{N-1} \alpha_{j,i} d_j$.

Substituting $\sum_{j=1}^{N-1} \alpha_{j,i} d_j = u_i - a$ in the previous equation gives $\hat{u}_i = u_i + (\lfloor a \rfloor - a)$. But, $-1 < \lfloor a \rfloor - a < 0$

$\lceil \hat{u}_i \rceil = \lceil u_i + (\lfloor a \rfloor - a) \rceil = u_i$, since u_i is an integer. Hence, $\mathbf{v} = \lfloor \mathbf{D}\mathbf{u} \rfloor$ and $\mathbf{u} = \left\lceil \mathbf{D}^{-1} \lfloor \mathbf{v} \rfloor \right\rceil$ define a GRIT.

2.3. Example

If $N = 4$, and each element of \mathbf{u} is subtracted from the next element to calculate the difference elements of \mathbf{v} , then one possible value of \mathbf{D} is given by:

$$\mathbf{D} = \begin{bmatrix} a_0/c & a_1/c & a_2/c & a_3/c \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

where $c = \sum_{i=0}^{N-1} a_i$

$$\mathbf{D}^{-1} = \begin{bmatrix} 1 & -(c-a_0)/c & -(a_2+a_3)/c & -a_3/c \\ 1 & a_0/c & -(a_2+a_3)/c & -a_3/c \\ 1 & a_0/c & (a_0+a_1)/c & -a_3/c \\ 1 & a_0/c & (a_0+a_1)/c & (c-a_3)/c \end{bmatrix}$$

In this case, the generalized, integer transform can be written as:

$$v_0 = \left\lfloor \frac{a_0 u_0 + a_1 u_1 + a_2 u_2 + a_3 u_3}{a_0 + a_1 + a_2 + a_3} \right\rfloor$$

$$v_1 = u_1 - u_0 \tag{1}$$

$$v_2 = u_2 - u_1$$

$$v_3 = u_3 - u_2$$

$$u_0 = v_0 - \left\lfloor \frac{(a_1 + a_2 + a_3)v_1 + (a_2 + a_3)v_2 + a_3 v_3}{a_0 + a_1 + a_2 + a_3} \right\rfloor$$

$$u_1 = v_1 + u_0 \tag{2}$$

$$u_2 = v_2 + u_1$$

$$u_3 = v_3 + u_2$$

3. THE EMBEDDING AND DETECTION ALGORITHM

3.1. The Embedding Process

The embedding process starts by grouping the pixels in the image into vectors of size $N \times 1$. The grouping can be based on pixel proximity or based on a secret key. Each vector is then classified into one of three groups according to the values of its difference coefficients.

- The first group is the “non-changeable” group. It contains the vectors whose difference coefficients are sensitive to embedding and hence would cause overflow or underflow in the pixel domain when the inverse transform is calculated.
- The second group is the “expandable” group. It contains the vectors whose difference coefficients can be expanded by multiplying them by two and then embedding a bit in the LSB without causing overflow or underflow in the pixel domain when the inverse transform is calculated.
- The third group is the “changeable” group. It contains the vectors whose difference coefficients can be changed by replacing their LSB without causing overflow or underflow in the pixel domain when the inverse transform is calculated.

The locations of all the vectors that are included in the expandable group are indicated by 1's in a binary map. We will refer to this map as the locations map. The JBIG algorithm is used to compress the map and reduce its size. Next, the LSBs of the difference coefficients of the vectors in the changeable group are collected into a bitstream. Next, the data to be hidden in the image is concatenated to this bitstream, and the result is concatenated to the compressed locations map to form the watermark data. Finally, this data is embedded in the difference coefficients of the vectors according to the type of the group they belong to.

No watermark bits are added to the non-changeable vectors, but one bit is embedded in each difference coefficient of a vector that belongs to the expandable or changeable group. If the vector belongs to the expandable group, a bit is embedded into a difference coefficient by first multiplying the coefficient by 2, then embedding the bit as the LSB of the resulting coefficient. If the vector belongs to the changeable group, a bit is embedded in the difference coefficient, simply by replacing the LSB.

3.2. Overflow and Underflow

The overflow and underflow due to the embedding process can be prevented by classifying the image vectors properly into the three groups described above. A vector is classified as expandable, if each of its difference coefficients (in the GRIT domain) can be multiplied by 2 and the LSB of each of the resulting difference coefficients can be replaced by an arbitrary bit without causing an overflow or underflow in the pixel domain.

Similarly, a vector is classified as changeable, if each of the LSBs of each of its difference coefficients (in the GRIT domain) can be replaced by an arbitrary bit without causing an overflow or underflow in the pixel domain.

Those vectors that cause overflow or underflow are classified as non-changeable.

3.3. Recursive Embedding

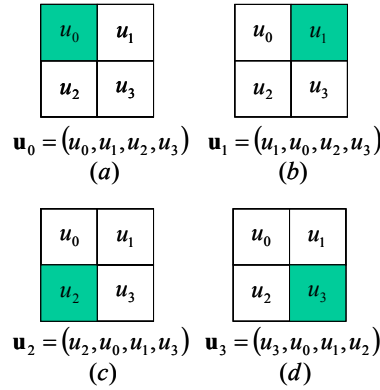


Fig. 1. Quad permutation for recursive embedding.

Since the embedding process is completely reversible, the algorithm can be applied to the image recursively to embed more data. However, the difference between the original image and the embedded image increases with every application of the algorithm. At some point, this difference becomes unacceptable for the intended application. One way to reduce the error when the algorithm is applied recursively is to use permutations of the entities of the input vector. For example, Fig. 1 depicts four different quad structures that can be used to permute a quad (4×1 vector). Each structure can be used in a different iteration for a total of four iterations. When \mathbf{u}_0 structure is used, the difference expansion is computed based on u_0 . Hence, the closer u_0 is to u_1 , u_2 , and u_3 , the smaller the difference is. Similarly, for \mathbf{u}_1 , \mathbf{u}_2 , and \mathbf{u}_3 , the difference expansion will be based on u_1 , u_2 , and u_3 , respectively. The complete recursive process with vector permutation is depicted in Fig. 2.

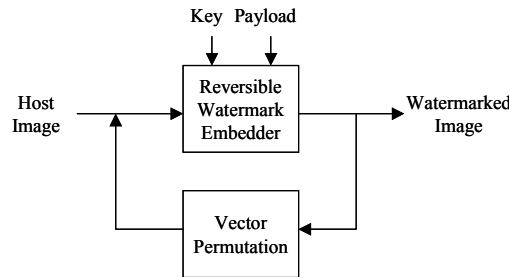


Fig. 2. Recursive embedding of the reversible watermark.

3.4. The Retrieval Process

The retrieval process of the embedded data starts by identifying all the vectors in the expandable group. This identification can be achieved by identifying all the changeable vectors in the embedded image. These vectors include all original expandable and changeable vectors.

The LSBs of the difference coefficients of these vectors represent the embedded watermark data. Hence, the LSB of each difference coefficient is collected to form the stream of embedded data. Next, the JBIG algorithm is used to decompress the first portion of this retrieved stream, which forms the locations map. This map identifies the locations of all the expandable vectors, and hence, separates the expandable vectors from the changeable vectors.

The original host image (before embedding) can now be recovered by dividing each difference coefficient in the expandable group by 2, rounding down the result, and replacing the LSB of each difference coefficient in the changeable group with the original bit taken from the retrieved watermark data. The embedded payload becomes available at the end of the host image retrieval process. It is, simply, the remaining bits in the bitstream.

4. EXPERIMENTAL RESULTS

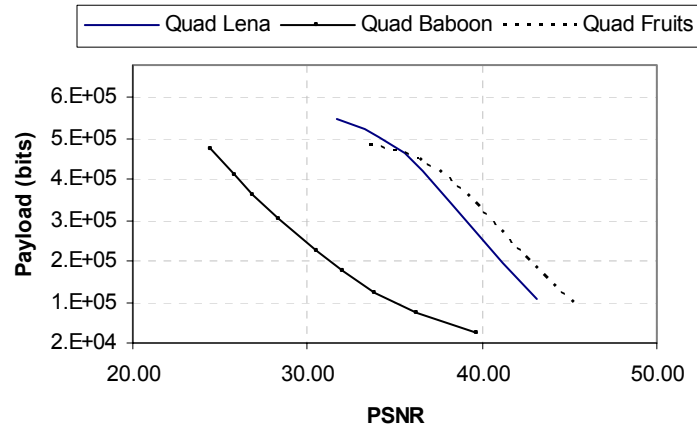


Fig. 3. Embedded payload size vs. PSNR for colored images embedded using non-recursive, quad-based algorithm.

We implemented the algorithm detailed in Section 3 and tested it with quads. Each quad was assembled from 2×2 adjacent pixels in the same color component as shown in Fig. 1a. We used a random binary sequence derived from uniformly distributed noise as a watermark signal. We tested the algorithm with the 512×512 RGB images: *Lena*, *Baboon*, and *Fruits*. We applied the algorithm to each color component non-recursively (single pass) and recursively (double pass). The payload size embedded into each of the test images (all color components) for the non-recursive application is plotted against PSNR in Fig. 3. The plot indicates that the achievable embedding capacity depends on the nature of the image itself. The algorithm performs with *Fruits* and *Lena* much better than *Baboon*, and it performs slightly better with *Fruits* than with *Lena*. With *Fruits*, the algorithm is able to embed 504 kB (1.92 bits/pixel) with image quality of 33.51 dB. It is also able to embed 202 kB (0.77 bits/pixel) with high image quality of 43.09 dB. Nevertheless, with *Baboon* the algorithm is able to embed 493 kB (1.87 bits/pixel) at 24.50 dB and 92 kB (0.35 bits/pixel) at 36.29 dB.

The visual quality of the watermarked images is shown in Fig. 4 and Fig. 5 for *Lena* and *Baboon* embedded at low, medium, and high PSNR. In general, the embedded images hardly can be distinguished from the original. However, a sharpening effect can be observed when the original and the embedded images are displayed alternatively. This effect is more noticeable at lower PSNR than at higher PSNR. It is also more noticeable for a high frequency image, such as *Baboon*, than for *Lena* and *Fruits*.

The payload size embedded into each of the test images (all color components) for the recursive application is plotted against PSNR in Fig. 6. Again, the plot indicates that the achievable embedding capacity depends on the nature of the image itself. The algorithm performs with *Fruits* and *Lena* much better than *Baboon*, and it performs slightly better with *Fruits* than with *Lena*. With *Fruits*, the algorithm is able to embed 982 kB (3.74 bits/pixel) with image quality of 28.42 dB. It is also able to embed 296 kB (0.77 bits/pixel) with high image quality of 39.05 dB. Nevertheless, with *Baboon* the algorithm is able to embed 808 kB (3.08 bits/pixel) at 20.18 dB and 130 kB (0.50 bits/pixel) at 32.62 dB.



Fig. 4. *Lena* embedded using non-recursive, quad-based algorithm (a) original, (b) 31.72 dB embedded with 569,838 bits, (c) 36.61 dB embedded with 441,794 bits, (d) 43.18 dB embedded with 130,001 bits.

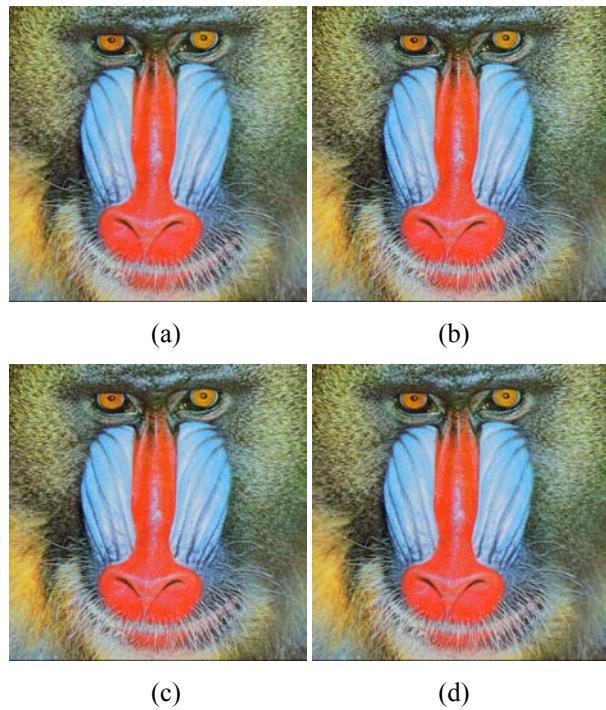


Fig. 5. *Baboon* embedded using non-recursive, quad-based algorithm (a) original, (b) 24.50 dB embedded with 492,708 bits, (c) 30.53 dB embedded with 244,439 bits, (d) 39.68 dB embedded with 43,113 bits.

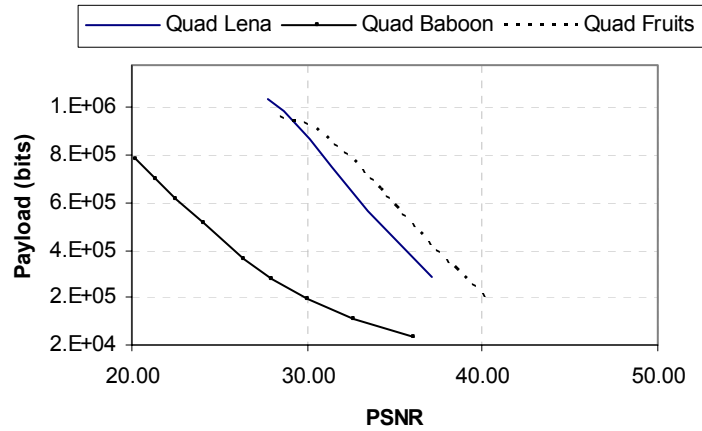


Fig. 6. Embedded payload size vs. PSNR for colored images embedded using recursive quad-based algorithm.

The visual quality of the watermarked images is shown in Fig. 7 and Fig. 8 for *Lena* and *Baboon* embedded at low, medium, and high PSNR. In general, the embedded images hardly can be distinguished from the original. However, a sharpening effect can be observed when the original and the embedded images are displayed alternatively. This effect is more noticeable at lower PSNR than at higher PSNR. It is also more noticeable for a high frequency image, such as *Baboon*, than for *Lena* and *Fruits*.



Fig. 7. *Lena* embedded using recursive, quad-based algorithm (a) original, (b) 28.61 dB embedded with 1,003,843 bits, (c) 33.48 dB embedded with 588,880 bits, (d) 37.15 dB embedded with 309,567 bits.

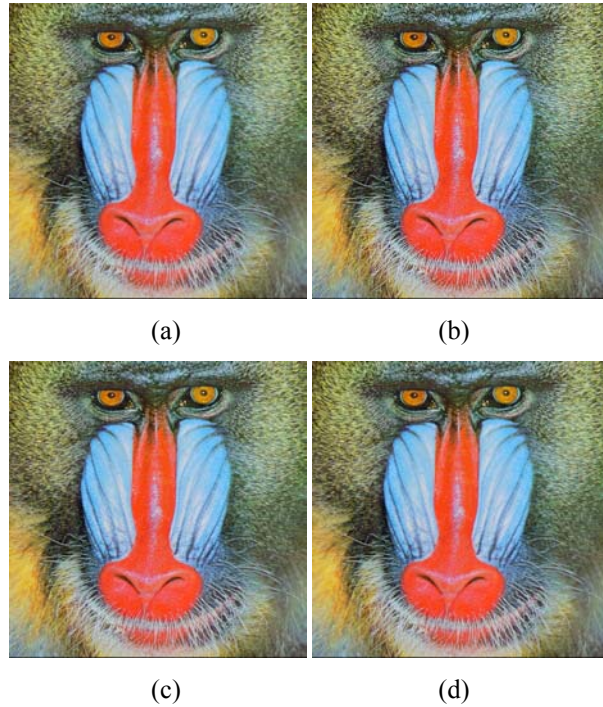


Fig. 8. *Baboon* embedded using recursive, quad-based algorithm (a) original, (b) 20.19 dB embedded with 807,710 bits, (c) 28.04 dB embedded with 301,827 bits, (d) 36.10 dB embedded with 49,963 bits.

Figure 9 compares the performance of the non-recursive, quad-based algorithm applied to the image once with that of the spatial, triplet-based algorithm described in [11]. This figure reveals that the quad-based and the triplet-based algorithms seem to have different operation ranges with some overlap. At higher PSNR, the triplet-based algorithm was unable to generate many results, but it can be observed from the tendency of the curves that the quad-based algorithm seems to have superior performance than the triplet-based algorithm. This result is because 2×2 quads have higher correlation than 1×3 triplets and because the single location map used by the quad-based algorithm is smaller than each of the two location maps used by the triplet-based algorithm (one location map for each pass).

On the other hand, although the quad-based algorithm was unable to generate many results at lower PSNRs, it can be observed from the tendency of the curves that the triplet-based algorithm seems to have superior performance. This behavior is attributed to the fact that the triplet-based algorithm is applied to the image twice (row-wise and column-wise), while the quad-based algorithm is applied only once.

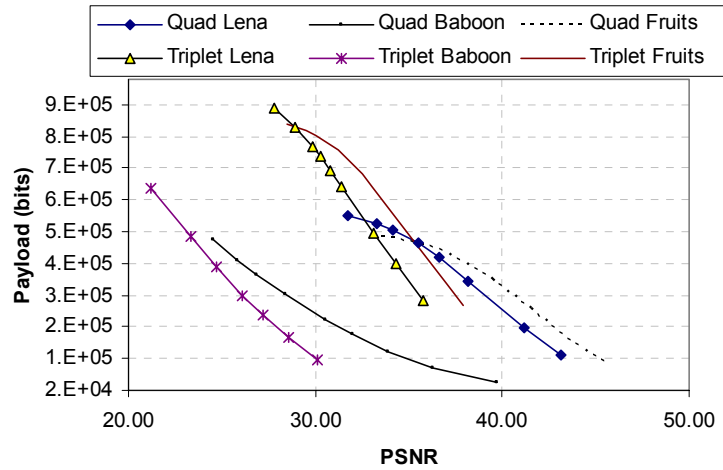


Fig. 9. Comparison between the performance of the triplet-based algorithm and the non-recursive, quad-based algorithm.

Figure 10 compares the performance of the recursive quad-based algorithm applied to the image twice with that of the spatial, triplet-based algorithm described in [11]. The figure reveals that the quad-based algorithm can embed about 100 kB more than the triplet-based algorithm at the same PSNR. The figure also shows that the quad-based algorithm can embed the same amount of as the triplet-based algorithm while producing images of about 2 dB higher in quality.

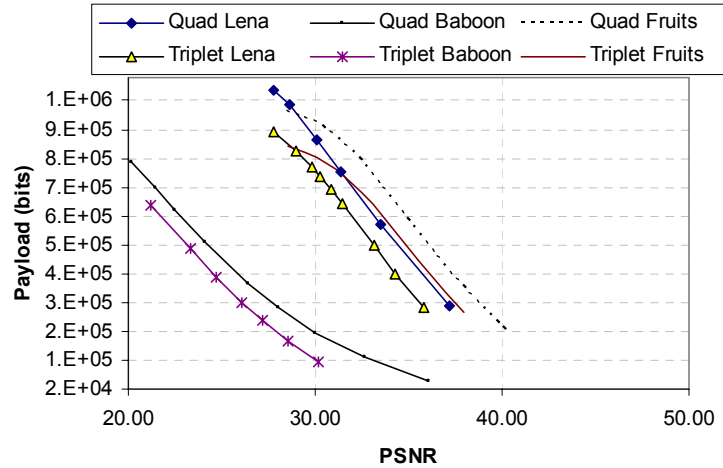


Fig. 10. Comparison between the performance of the triplet-based algorithm and the recursive, quad-based algorithm.

5. CONCLUSIONS

In this paper, we present a high-capacity, invertible, data-hiding algorithm suitable for watermarking valuable and sensitive images such as original novel artworks and medical and military images. The algorithm lets the user recover the original host image after he or she retrieves the hidden data. The algorithm can easily embed the large amount of data typically associated with artworks and medical and military images without degrading the image quality. The algorithm is based on a generalized, reversible, integer transform. The test results indicate that the performance of the quad-based algorithm is superior to that of the spatial, triplet-based algorithm at higher PSNR.

REFERENCES

1. F. Mintzer, J. Lotspiech, and N. Morimoto. (1997 Dec.). Safeguarding digital library contents and users: digital watermarking. *D-lib magazine*. [Online]. Available: [http://www. Dlib.org](http://www.Dlib.org).
2. C. W. Honsinger, P. W. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," U.S. Patent 6,278,791, 2001.
3. B. Macq, "Lossless multiresolution transform for image authenticating watermark," in *Proc. of EUSIPCO*, Tampere, Finland, Sept. 2000.
4. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313-336, 1996.
5. C. De Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of histogram for reversible watermarking," in *Proc. of IEEE 4th Workshop on Multimedia Signal Processing*, Oct. 2001.
6. J. Fridrich, M. Goljan, and Rui Du, "Invertible authentication," in *Proc. SPIE Photonics West, Security and Watermarking of Multimedia Contents III*, vol. 3971, San Jose, California, Jan. 2001, pp. 197-208.
7. J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding—new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 2, pp. 185-196, Feb. 2002.
8. J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding for all image formats," in *Proc. SPIE Photonics West, Electronic Imaging 2002, Security and Watermarking of Multimedia Contents*, vol. 4675, San Jose, California, Jan. 2002, pp. 572-583.
9. M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible data hiding," in *Proc. of the IEEE International Conference on Image Processing*, vol. II, Sept. 2002, pp. 157-160.
10. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. on Image Processing*, vol. 12, no. 8, August 2000, pp. 890-896.
11. A. M. Alattar, "Reversible Watermark Using Difference Expansion of Triplets," in *Proc. of the 2003 IEEE International Conference on Image Processing, ICIP'2003*, Barcelona, Spain, September 14-17, 2003, pp. 501-504.
12. T. Kalker and F.M. Willems, "Capacity bounds and code constructions for reversible data-hiding," in *Proc. of Electronic Imaging 2003, Security and Watermarking of Multimedia Contents V*, Santa Clara, California, Jan. 2003.
13. T. Kalker and F. M. Willems, "Capacity bounds and code constructions for reversible data-hiding," in *Proc. of the International Conference on Digital Signal Processing*, June 2002, pp.71-76.