Reversible Data Hiding Giving Priority to the Use of Edges and Textures

YONGJIAN HU[1,2], HEUNG-KYU LEE[1], JIANWEI LI[2]

[1]Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology, Daejeon, Korea
[2]College of Automation Science and Engineering
South China University of Technology, Guangzhou, P.R. China
Email: hklee@mmc.kaist.ac.kr; eeyjhu@scut.edu.cn

Keywords: digital watermarking, reversible data hiding, lossless watermark, difference expansion embedding, JBIG compression.

Abstract

We propose a new difference expansion (DE) embedding based reversible data hiding algorithm for digital images. Unlike other current DE-based algorithms that are primarily interested in using small pixel-pair differences (i.e., flat image regions) for data hiding, our algorithm gives priority to the use of image edges and textures. The experiments prove that our algorithm can produce visual image quality more suitable for human visual characteristics.

1 Introduction

Difference expansion embedding is a popular technique for reversible data hiding/watermarking algorithms to achieve large embedding capacity. Most of DE-based reversible data hiding algorithms (e.g.: [1-5]) prefer choosing small pixel-pair differences for embedding. For example, Tian [1] first selected the low-frequency coefficients of integer Haar wavelet transform (i.e., image pixel-pair differences) with small magnitude for DE expansion embedding. Alattar [4] extended Tian's pixel-pair difference expansion method using difference expansion of vectors. Kamstra et al. [5] improved Tian's method and selected embeddable differences using a sorting list based on the characteristics of the low-pass image. Small differences tend to occur at the beginning of the sorting list. In addition to the methods developed in integer Haar wavelet transform domain, some researchers also proposed DE expansion methods in other domains. For example, to better use the correlation information of neighboring pixels, Thodi et al. [2] used image prediction rather than integer Haar wavelet transform. They still gave priority to small predicted pixel errors for DE expansion embedding. Thodi et al. further proposed a histogram-based selection scheme for choosing small differences in [3]. The reason of giving priority to small differences for DE expansion embedding is to acquire a high peak-signal-to-noise ratio (PSNR) value of the embedded image. In the aforementioned algorithms, PSNR is used as a metric to evaluate visual quality of watermarked images. However, the PSNR value is essentially the measurement of statistical errors of a modified image rather than a metric for visual perception of the human eye. On the other hand, small differences usually correspond to flat image regions. The alteration to those differences would make embedding distortion first appear in homogeneous image regions, to which the human eye is very sensitive.

According to knowledge of human visual systems, image textures have a substantial masking effect to the human eye and the human visual systems are not sensitive to the change in textures. Besides, image edge is able to bear alteration to some degree [6]. So we propose a new reversible data hiding algorithm that begins watermark/data embedding from image edges and textures. The experiments demonstrate that visual quality of our watermarked images is more suitable for human visual characteristics.

2 Framework of the proposed algorithm

Our data embedding is based on differences. As stated above, there are different ways to calculate differences in image processing (e.g.: [1, 2]). The proposed method can be extended to different types of differences. For simplicity, we adopt the predictor in [2] to calculate differences, which are basically predicted image errors.

$$\hat{x} = \begin{cases} \max(a,b) & \text{if } c \le \min(a,b) \\ \min(a,b) & \text{if } c \ge \max(a,b) \\ a+b-c & \text{otherwise} \end{cases} \quad (1)$$

where x and $\hat{x}$ are the current pixel and its predicted value, respectively. a, b and c are the south, east, and southeast neighbors, respectively. The predicted pixel error, p_e, is computed by $p_e = x - \hat{x}$. For generality, we sometimes call p_e as the difference. For simplicity of description, we further define the following notations that will be used in this paper. Let I, I', $\hat{I}$, and I_e denote the original image, watermarked/embedded image, predicted image, and predicted error image, respectively. Let P and Q denote the pure payload and the header file, respectively. Let M and M denote the overflow location map and its JBIG compressed version, respectively. Let $\eta(\cdot)$ be the function to calculate the length of a bitstream (unit: bit). Let *count* represent the

number of selected embeddable predicted errors. $[-T_{nl},-T_{nr}]$ and $[T_{pl},T_{pr}]$ are the embedding zones in the negative and positive parts of the horizontal axis, respectively. Here, $T_{nl} \geq T_{nr} > 0$ and $T_{pr} \geq T_{pl} \geq 0$. $p_{e_{exml}}$ and $p_{e_{exmr}}$ are the left and right ends of the histogram, respectively.

Generally, the histogram of predicted errors follows a zero-mean Laplacian distribution, as shown in Fig.1. On the other side, according to knowledge of image processing, predicted errors with large magnitude usually correspond to image edges; predicted errors with middle magnitude correspond to textures; whereas, predicted errors with small magnitude correspond to flat regions. Since our algorithm gives priority to the use of edges and textures, it means that we begin to search for embeddable predicted pixel errors from both ends of the histogram.

The framework of our algorithm is depicted in Fig. 2. We will describe our technical details in the next section.

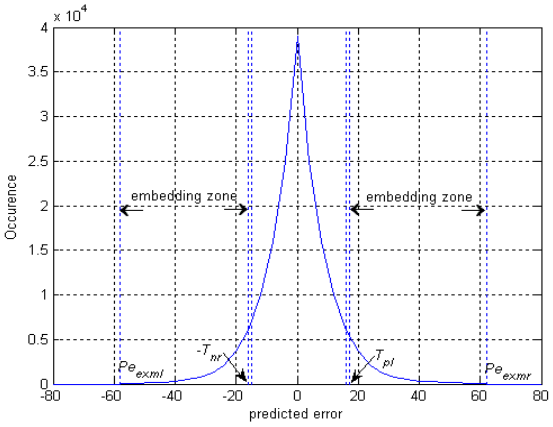

Fig. 1 Sample histogram of predicted errors.

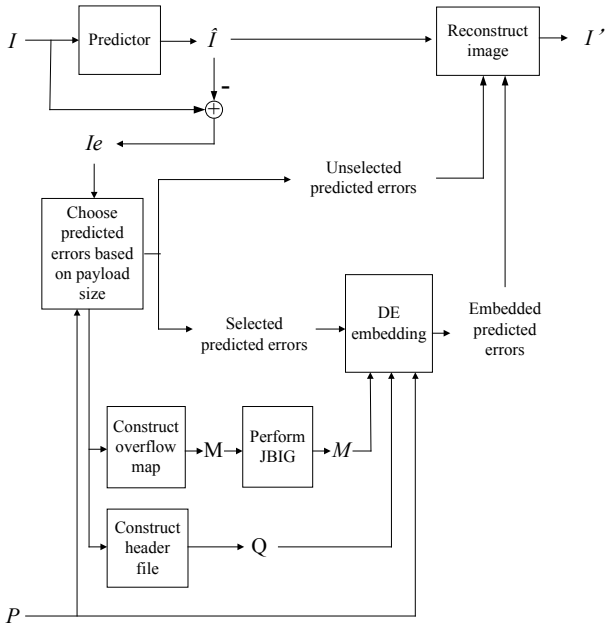

Fig. 2 Framework of our algorithm.

3 Algorithm

This section gives the details of our algorithm. We first discuss the embedding and extraction formulas, and then, show how to construct the overflow location map and the header file. Finally, we list the steps to implement our algorithm in a predicted error image.

3.1 Embedding and extraction formulas

We embed one binary information bit i into a predicted error. p_e can be represented as $p_e = b_{l-1}b_{l-2}...b_0$, where l is the length of binary representation. The traditional DE expansion embedding rule [2] is

$$p_e^{'} = b_{l-1}b_{l-2}...b_0 i = 2p_e + i \qquad (2)$$

where $p_e^{'}$ is the embedded difference. The embedded pixel value, $x^{'}$, is computed by $x^{'} = \hat{x} + p_e^{'}$. Their corresponding data extraction and pixel value recovery formulas are

$$\begin{cases} i = | p_e^{'} | \%2 \\ p_e = \lfloor p_e^{'} / 2 \rfloor \\ x = x^{'} - p_e - i \end{cases} \qquad (3)$$

where $\%$ denotes the modulo operation, and $\lfloor \cdot \rfloor$ is the floor function. However, if we directly use (2) in our algorithm, the embedding distortion would be high. We make some explanation. The core idea of DE embedding is that the binary representation of a difference has to be left-shifted one bit to obtain the vacant LSB position for embedding; otherwise, the decoder does not know the location of the hidden bit in the process of blind data extraction. However, one-bit left-shifting operation doubles the absolute of the difference. For a difference with small magnitude, this left-shifting operation only brings limited distortion to the resulting difference; but for a difference with large magnitude, such an operation would bring a great impact on the result. Since we begin to select embeddable differences from the ends of the histogram, the first selected differences often have large magnitude. If we use (2), the original pixel values would be greatly distorted after embedding. Therefore, we try to decrease the distortion by using the following modified embedding formulas:

$$\begin{cases} p_e \in [-T_{nl},-T_{nr}] : p_e^{'} = -T_{nr} + 2(p_e + T_{nr}) - i = 2p_e + T_{nr} - i \\ p_e \in [T_{pl},T_{pr}] : p_e^{'} = T_{pl} + 2(p_e - T_{pl}) + i = 2p_e - T_{pl} + i \end{cases} \qquad (4)$$

where $[-T_{nl},-T_{nr}]$ and $[T_{pl},T_{pr}]$ are respectively the left and right embedding zones, as shown in Fig. 1. After embedding, the altered values fall into $[-2T_{nl} + T_{nr} - 1,-T_{nr}]$ and $[T_{pl},2T_{pr} - T_{pl} + 1]$. These two intervals indicate that the embedding distortion is much smaller than that by directly using (2).

Their corresponding data extraction and pixel value recovery formulas can be described in the following two cases:

- If $p_e^{'} \in [-2T_{nl} + T_{nr} - 1, -T_{nr}]$ and $x^{'}$ has a position indicated as "0" (i.e., embeddable, as will be discussed in next subsection) in the overflow location map, we have

$$\begin{cases} i = | p_e^{'} - T_{nr} | \%2 \\ p_e = -\lfloor | p_e^{'} - T_{nr} | / 2 \rfloor \\ x = x^{'} - p_e - i \end{cases} \qquad (5)$$

- If $p_e^{'} \in [T_{pl}, 2T_{pr} - T_{pl} + 1]$ and $x^{'}$ has a position indicated as "0" in the overflow location map, we have

$$\begin{cases} i = | p_e^{'} + T_{pl} | \%2 \\ p_e = \lfloor | p_e^{'} + T_{pl} | / 2 \rfloor \\ x = x^{'} - p_e - i \end{cases} \qquad (6)$$

The construction of the overflow location map will be explained next.

3.2 Overflow location map and the header file

Applying (4) to a predicted error may cause the embedded pixel value out of the range [0, 255] for an 8-bit image, even the predicted error is in the embedding zone. The overflow/underflow problem occurring at those pixels makes lossless recovery of the original image impossible. So we have to avoid using those pixels. In this paper, a 2-D binary overflow location map, M, is introduced to indicate possible overflow locations. M is initialized by "0". All of possible overflow locations, where the pixel value can not satisfy the following constraint, are indicated as "1s".

$$\begin{cases} p_e \in [-T_{nl}, -T_{nr}] : \hat{x} + (2p_e + T_{nr} - 1) < 0 \\ p_e \in [T_{pl}, T_{pr}] : \hat{x} + (2p_e - T_{pl} + 1) > 255 \end{cases} \qquad (7)$$

A predicted error or pixel is called as an un-embeddable one if (7) is not satisfied; otherwise, it is an embeddable one.

For data extraction and image recovery, we need not only the overflow location map but also the header file, Q, which records all embedding information. In this paper, Q consists of T_{nl} (8 bits), T_{nr} (8 bits), T_{pl} (8 bits), T_{pr} (8 bits), the coordinate of the last embedded pixel in the image (32 bits), and the length information of JBIG-compressed overflow location map, M.

3.3 Implementation

Data hiding is implemented through the two processes: selecting the embedding zones and embedding on the selected zones. These two processes are realized in two separate loops. In the first process, we begin to select embeddable predicted errors from the histogram ends. Since the histogram usually has two tails with different lengths, we do it from the histogram side with the longer tail. The selection on that side continues until the histogram has two balanced tails. Afterwards, we begin to interleavingly select embeddable predicted errors from the two sides. The use of interleaving selection is to select just necessary number of embeddable pixels so as to avoid raising unnecessary embedding distortion. We give the complete data hiding steps as follows.

- **Step 1:** Perform (1) on the original image from its upper left corner and in a raster scanning manner to obtain the predicted error image, and then, calculate the predicted error histogram.
- **Step 2:** Begin the first loop to determine the embedding zone. Assume $| p_{e_{exml}} | \geq | p_{e_{exmr}} |$. We select embeddable predicted errors from the left side of the histogram. Let $-T_{nl} = -T_{nr} = p_{e_{exml}}$ and $T_{pl} = T_{pr} = p_{e_{exmr}}$. The initial embedding zone is $[-T_{nl}, -T_{nr}]$. We test each predicted error located in the embedding zone by (7) and construct the overflow location map. After each iteration, we judge whether we get enough embeddable predicted errors for the payload by the following formula:

$$count \geq \eta(P) + \eta(Q) + \eta(M) \qquad (8)$$

  If (8) is satisfied, stop the current search loop and go to Step 3; otherwise, decrease T_{nr} by 1 (i.e., the increase of width of embedding zone) and repeat the search loop. We recalculate *count* in the new embedding zone and judge whether (8) is satisfied. If $T_{nr} = p_{e_{exmr}}$ and (8) is still not satisfied, we begin the aforementioned interleaving selection. The embedding zones become $[-T_{nl}, -T_{nr}] \cup [T_{pl}, T_{pr}]$. We interleavingly decrease T_{nr} and T_{pl} by 1, and calculate *count* and test (8). The search loop continues until (8) is satisfied.
- **Step 3:** In the second process, we embed P, Q and M into the original image according to M. Data embedding loop begins from the upper left corner of the image and in a raster scanning manner. However, to facilitate data decoding, we borrow the idea of [2] and embed the bitstream of Q and M in the LSBs of first $\eta(Q) + \eta(M)$ pixels of the watermarked image while the original $\eta(Q) + \eta(M)$ LSBs are saved in the place previously allotted for Q and M. Such an exchange of storage places does not affect the algorithm performance. Detailed manipulation can be found in [2].
- **Step 4:** As soon as the data embedding process is finished, the watermarked image is created.

At the decoder, we get back Q and M from the LSBs of first $\eta(Q) + \eta(M)$ pixels of the test image before performing data extraction. With Q and M, we can extract the embedded data in the exact reverse manner of watermark embedding. We use (5) and (6) to get back the embedded data and resume the original pixel value. As soon as the data extraction process is finished, we also obtain the losslessly recovered original image.

4 Experimental results

We test our algorithm on different types of images. Some experimental results are given in Figs. 3 and 4. For the sake of comparison, we also give the results of Tian's DE algorithm [1] and Thodi et al.'s P2 algorithm [3].

According to Fig. 3, Thodi et al.'s algorithm has the best PSNR values under all embedding rates. Tian's algorithm has higher PSNR values than ours, especially, at low and middle embedding rates. However, due to using the embedding formula (4), the performance of embedding rate vs. PSNR curve of our algorithm has the slowest decrease in PSNR as the payload increases. Besides, at the largest embedding rate, our algorithm has the same PSNR value as Thodi et al.'s P2 algorithm, which is higher than the PSNR value of Tian's DE algorithm. Fig. 4 demonstrates the advantages of our algorithm in visual quality. We compare the amplified parts of the watermarked images at typical embedding rates (e.g.: 0.7 bpp). For Lena, (a) has the poorest visual quality. It can be seen that blocking effects are especially annoying in edges and flat regions. On the other hand, (b) looks a little blurred. In particular, the edges (e.g., hat edges and brows) are not clear. However, (c) obtained from our algorithm has the best visual appearance. It has smooth flat regions, clear edges and vivid textures. For F-16, the situation is similar to what occurs in Lena. For example, (a) has apparent artifacts in flat regions and edges. (b) looks blurred in edges, star and numbers. (c) has clear edges, star and numbers, and its flat regions look very natural. The above experiments demonstrate that, although our PSNR value is not the highest, our algorithm produces the best visual quality among the three algorithms. Practically, applying (2) to edges and textures is similar to sharpening these image regions first. Moreover, letting flat regions to be the last embedding zone is a good way to avoid raising suspicion of content alteration at first glance.

5 Conclusion

We propose a new algorithm for reversible data hiding. Unlike traditional methods that aim at low PSNR values and begin data embedding from flat image regions, the proposed method gives priority to hiding data on edges and textures. Under the same embedding rate, our algorithm yields better visual appearance than other typical algorithms. Besides, it does not sacrifice the maximum embedding capacity.

Acknowledgements

This work was supported in part by KOSEF grant NRL program R0A-2007-000-20023-0 of MOST, NSF of China 60772115, 60572140, and NSF of Guangdong 04020004.

References

[1] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890-896, Aug. 2003.

[2] D.M. Thodi, and J.J. Rodriguez, "Prediction-error based reversible watermarking," in *Proc. of IEEE Int. Conf. on Image Processing*, vol. 3, pp. 1549-1552, Oct. 2004.

[3] D.M. Thodi, and J.J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 721-730, Mar. 2007.

[4] A.M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. on Image Processing*, vol. 13, no. 8, pp. 1147-1156, Aug. 2004.

[5] L. Kamstra, H.J.A.M. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Trans. on Image Processing*, vol. 14, no. 12, pp. 2082-2090, Dec. 2005.

[6] A. B. Watson, *Digital Images and Human Vision*. Cambridge, MA: MIT Press, 1993.

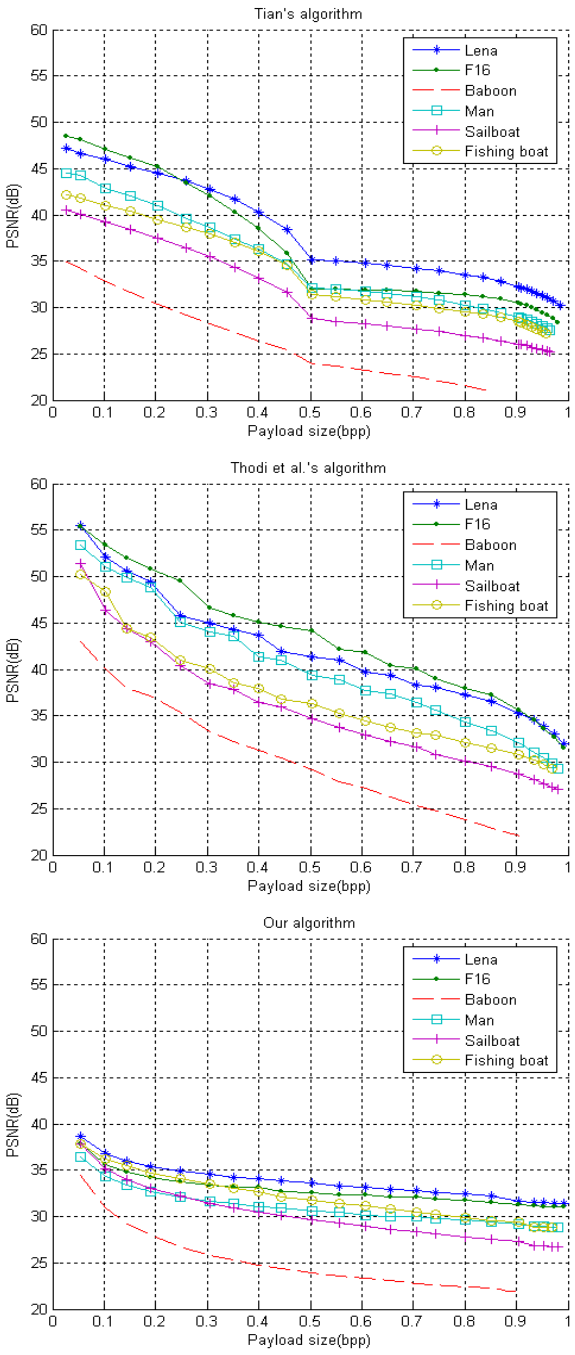

Fig. 3 Embedding rate versus PSNR curves.

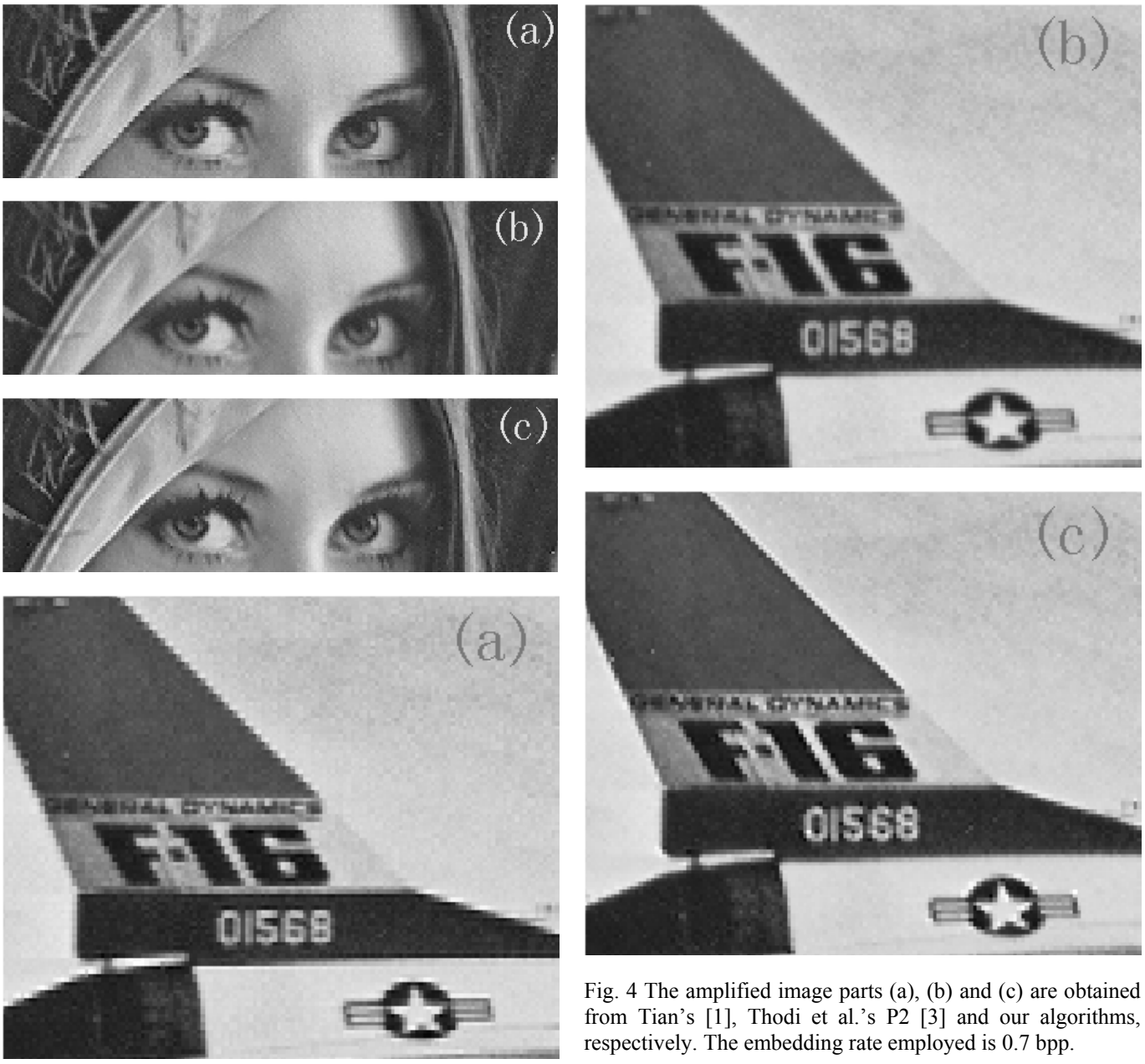

Fig. 4 The amplified image parts (a), (b) and (c) are obtained from Tian's [1], Thodi et al.'s P2 [3] and our algorithms, respectively. The embedding rate employed is 0.7 bpp.