Applied Mathematics and Computation xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Capacity maximization for reversible data hiding based on dynamic programming approach ☆

Kuo-Liang Chung [a,*], Yong-Huai Huang [a], Wei-Ning Yang [b], Yu-Chiao Hsu [b], Chyou-Hwa Chen [a]

[a] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC

[b] Department of Information Management, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC

ARTICLE INFO

Keywords:
Embedding capacity maximization
Dynamic programming
Histogram modification
PSNR
Reversible data hiding

ABSTRACT

Recently, an efficient reversible lossless data hiding algorithm by Ni et al. was presented. Their fast algorithm can recover the original image without any distortion and its PSNR lower bound is higher than that of all existing reversible data hiding algorithms. Instead of selecting the peak-valley pairs in a greedy way, this paper presents a dynamic programming-based reversible data hiding algorithm to determine the most suitable peak-valley pairs such that the embedding capacity object can be maximized. Based on some artificial map images, experimental results demonstrate that our proposed algorithm has 9% embedding capacity improvement ratio and has the similar image quality performance when compared to Ni et al.'s algorithm although it has some execution-time degradation. For natural images, the embedding capacity of Ni et al.'s algorithm is very close to the maximal embedding capacity obtained by our proposed algorithm. Furthermore, the comparison between our proposed dynamic programming-based algorithm and the reversible data hiding algorithm by Chang et al. is investigated.

1. Introduction

Data hiding is an important technique for authentication, identification, annotation, and copyright protection [8]. In the past decades, many types of data hiding algorithms have been developed, such as the least significant bit plane (LSB)-based algorithm [14,16,24,5], the spread-spectrum-based data hiding algorithm [9,23,21,11,18,19], the singular value decomposition (SVD)-based algorithm [17,6,7,3], and the vector quantization (VQ)-based scheme [15,2]. Among these developed data hiding algorithms, the hiding data can be embedded into images without causing visual degradation. However, once the original image has been modified by the data hiding algorithm, it is very hard to recover the original image from the marked one when the receiver demands to use the original image.

In order to reverse the marked image back to the original one, some reversible data hiding algorithms have been developed to achieve the reversibility goal for the marked image. Based on the modulo 256 addition, Honsinger et al. presented the first reversible algorithm [13] to embed data into gray images. By using the lossless multiresolution transform, Macq and Deweyand [20] decomposed the original image into several subbands and applied the modulo 256 addition to embed data into these decomposed subbands. In [10], Fridrich et al. first compressed some selected bit planes of the original image, and then both the hidden data and the compressed bit planes were embedded into the selected bit planes. Due to concentrating

☆ Supported by National Council of Science of R.O.C. under contracts NSC96-2221-E-011-027 and NSC97-2221-E-011-128.

* Corresponding author.
E-mail address: k.l.chung@mail.ntust.edu.tw (K.-L. Chung).

0096-3003/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2008.10.066

on the authentication, the above three reversible data hiding algorithms only provide limited data hiding capacity, i.e. limited embedding capacity.

To increase the capacity of the reversible data hiding, Goljan et al. [12] segmented the original image into non-overlapping blocks and classified each block into two types. The hidding data is embedded by flipping a block from one type to the other type and the original type of each block is also embedded for reversibility. Based on the integer wavelet transform, Xuan et al. [25] embedded the data into the selected bit planes of wavelet coefficients in middle and high frequency subbands. The binary values of selected bit planes are first compressed losslessly, and then they are embedded into the original image together with the hiding data. Celik et al. [1] quantized gray values and then by using the predictive coding, these quantized gray values were used as the side information to losslessly compress the quantization residuals. Both the compressed residuals and the hiding data were embedded into the LSB of the original image. In the data extracting process, the original image can be recovered by adding decompressed quantization residuals to quantized gray values in the marked image. Chang et al. [4] presented a reversible data hiding algorithm based on the predictive coding. In the data embedding process, the predictive coding is first performed to predict the gray value of each pixel, and then the hiding data can be embedded by modifying the these predicted gray values. In the data extracting process, the predictive coding is performed again to extract the embedded data and restore the original gray value of each pixel. Although the above four reversible data hiding algorithms [12,25,1,4] have higher embedding capacity, the peak signal to noise ratios (PSNRs) between the original image and the marked one for the four algorithms are some degraded and they are time-consuming.

Recently, Ni et al. [22] presented a novel reversible data hiding algorithm based on the histogram modification technique. Using the greedy approach, the histogram of the original image is first segmented into several non-overlapped intervals and each interval is bounded by a peak-valley pair. The key idea in Ni et al. [22]'s algorithm is that the pixels contributed to the peak point of each peak-valley pair in the histogram can be used to embed data by slightly modifying their gray values. Experimental results showed that the capacity of the reversible data hiding can be significantly increased by embedding the data into peak points of the histogram. Especially, due to only modifying the gray values of concerned pixels slightly, the PSNR lower bound of Ni et al.'s algorithm is higher than any one of all existing reversible data hiding algorithms.

Instead of using the greedy approach in Ni et al.'s algorithm to select the set of peak-valley pairs heuristically, this paper presents a dynamic programming approach to determine the most suitable set of peak-valley pairs and the determined set can maximize the embedding capacity while has the similar image quality performance. Based on some artificial map images, experimental results demonstrate that our proposed algorithm has 9% embedding capacity improvement ratio and has the similar image quality performance when compared to Ni et al.'s algorithm although it has some execution-time degradation. For natural images, the embedding capacity of Ni et al.'s algorithm is very close to the maximal embedding capacity obtained by our proposed algorithm. Furthermore, the comparison between our proposed dynamic programming-based algorithm and the reversible data hiding algorithm by Chang et al. is investigated.

The rest of this paper is organized as follows. In Section 2, Ni et al.'s reversible data hiding algorithm is surveyed. In Section 3, our proposed dynamic programming-based algorithm is presented to maximize the embedding capacity of Ni et al.'s algorithm. In Section 4, some experiments are carried out to demonstrate the advantage of our proposed algorithm. Finally, some concluding remarks are addressed in Section 5.

2. Past work by Ni et al.

In Ni et al.'s algorithm, the data is embedded into the original image by slightly modifying its histogram. Assume that we have an original image I and its histogram $H(I)$ is shown in Fig. 1a. For ease of exposition, let $h(x)$ denote the number of pixels with gray value $x \in [0,255]$. In Fig. 1a, it can be observed that the gray value p is the peak point in $H(I)$ since $h(p) > h(x)$ for all $x \neq p$; that is, p is the most frequent gray value appeared in the original image I. In addition, gray value v is the valley point in

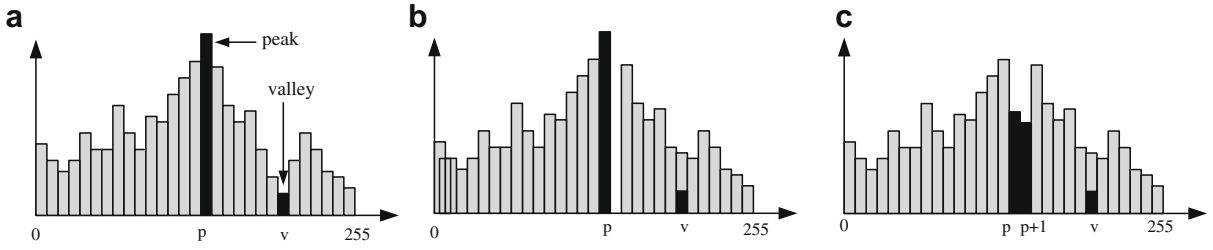

Fig. 1. The main idea in Ni et al.'s algorithm. (a) A histogram example with only one peak point and one valley point. (b) Shift the interval (p, v) to the right by one unit. (c) Embedding the data into the positions with gray levels p and $p+1$.

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

$H(I)$ since $h(v) < h(x)$ for all $x \neq v$. From the histogram $H(I)$ and the peak-valley pair $\langle p, v\rangle$, the hiding data, represented by the binary string B_h, can be embedded into the original image I by the following embedding procedure:

Step 1. If $h(v) > 0$, the positions of pixels with gray value v are transformed into a binary string B_v and then the hiding data B_h and B_v are combined to form a new binary string B'_h. The binary string B_v contains the overhead information for recovering the original image from the marked one.

Step 2. If $p < v$, shift the rectangular bars corresponding to the gray values within the interval $[p+1, v-1]$ in the histogram to the right by one unit by adding one to gray values of the pixels with gray values within $[p+1, v-1]$ (see Fig. 1b). If $p > v$, shift the rectangular bars corresponding to the gray values within $[v+1, p-1]$ in the histogram to the left by one unit by subtracting one from the gray values of the pixels with gray values within $[v+1, p-1]$.

Step 3. Scan the original image to select the pixels with gray value p to embed B'_h. For each selected pixel, its gray value remains p when the corresponding embedded bit in B'_h is 0 and is modified to $p+1$ (or $p-1$) when the corresponding embedded bit is 1 for the case of $p < v$ (or $p > v$) (see Fig. 1c).

From the above three steps, we can embed $h(v) - O_{B_v}$ bits of hiding data into the peak-valley pair $\langle p, v\rangle$ where O_{B_v} denotes the number of bits required in the binary string B_v. The condition $h(v) - O_{B_v} > 0$ holds because $h(p)$ is usually much larger than $h(v)$. Since the upper bound on the mean square error (MSE) between the original image and the marked image is 1, the PSNR lower bound of Ni et al.'s algorithm is 48.13 $\left(= 10\log_{10}\frac{255^2}{1^2}\right)$ and it is higher than those of all existing reversible data hiding algorithms.

Given a marked image, the following extracting procedure can be used to extract the hidden data and reverse the marked image back to the original one:

Step 1. Scan the marked image in the same order used in the embedding process. For each pixel examined, a bit 0 is extracted if the gray value is p and a bit 1 is extracted if the gray value is $p+1$ (or $p-1$) for the case of $p < v$ (or $p > v$). After examining all pixels, a binary string B'_h can be obtained to retrieve the hiding data string B_h and the overhead information string B_v.

Step 2. Scan the marked image again. for the case of $p < v$ (or $p > v$), decrease (or increase) the gray values of the pixels with gray values within $[p+2, v]$ ($[v, p-2]$).

Step 3. Transform the overhead information string B_v to locate the corresponding pixel positions. Reset the gray value of each located pixel to v to recover the original image.

Naturally, the embedding capacity can be enhanced by considering more than one peak-valley pair for embedding the hiding data. For the original image I, n valley points $v_1, v_2, \ldots, v_n$, where $0 < v_1 < v_2 < \cdots < v_n < 255$, are first selected from the histogram $H(I)$. Since the two gray values 0 and 255 are not used in Ni et al.'s algorithm, we let v_0 and v_{n+1} denote two pseudo valley points where $v_0 = 0$ and $v_{n+1} = 255$ for easy exposition; the two pseudo valley points are only used for explaining the peak-valley pair selection, instead of constructing the peak-valley pairs. In practical implementation, for each valley point v_i, $0 < i < n+1$, one of the two conditions $h(p_i) - h(v_i) * O_{v_i} > 0$ and $h(p_{i+1}) - h(v_i) * O_{v_i} > 0$ must be held where p_i and p_{i+1} denote two peak points in the intervals (v_{i-1}, v_i) and (v_i, v_{i+1}), respectively; O_{v_i} is the number of bits needed for recording the position of each pixel with gray value v_i and it is defined by $O_{v_i} = [\log_2(\max(W, H))] \times 2$ where W and H is the width and height of the input image. According to the following peak-valley pair selection procedure, n peak-valley pairs, $\langle p_1, v_1\rangle, \langle p_2, v_2\rangle, \ldots, \langle p_n, v_n\rangle$, can be determined in a greedy way to satisfy the memory requirement for the hiding data B_h and the overhead information string B_v:

Step 1. For two intervals (v_0, v_1) and (v_n, v_{n+1}), select two gray values $p_{(1)(0)}$ and $p_{(n)(n+1)}$, $v_0 < p_{(1)(0)} < v_1$ and $v_n < p_{(n)(n+1)} < v_{n+1}$, as two peak points.

Step 2. For each interval (v_i, v_{i+1}), $1 < i < n$, select two gray values $p_{(i)(i+1)}$ and $p_{(i+1)(i)}$, $v_i < p_{(i)(i+1)} < p_{(i+1)(i)} < v_{i+1}$, as two peak points.

Step 3. For each v_i, $1 < i < n$, if $h(p_{(i)(i-1)}) > h(p_{(i)(i+1)})$, set $p_i = p_{(i)(i-1)}$; otherwise, set $p_i = p_{(i)(i+1)}$. Consequently, n peak-valley pairs $\langle p_1, v_1\rangle, \langle p_2, v_2\rangle, \ldots, \langle p_n, v_n\rangle$ are determined.

After obtaining n peak-valley pairs, running the embedding procedure on each peak-valley pair can embed the hiding data B_h and the overhead information B_v into the original image. Similarly, applying the extracting process to each peak-valley pair can retrieve the hiding data B_h and recover the original image.

3. The proposed dynamic programming-based algorithm for maximizing embedding capacity

Since each peak-valley pair in Ni et al.'s reversible data hiding algorithm is determined by using the greedy approach, it may not achieve the maximal embedding capacity goal. In this section, the analysis is first provided to measure the maximal capacity of the histogram, and then our proposed dynamic programming-based data hiding algorithm is presented to determine the most suitable peak-valley pairs in order to maximize the embedding capacity and preserve the image quality.

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

3.1. Maximal embedding capacity of the histogram

Given the histogram $H(I)$ of the input image I with size $W \times H$, let $\langle x_0, x_1\rangle$, $x_0 < x_1$, denote a gray value-pair. The value-pair $\langle x_0, x_1\rangle$ can be viewed as a peak-valley pair $\langle p, v\rangle$ where $v = \min(h(x_0), h(x_1))$ and $p = \max(h(x_0), h(x_1))$. The embedding capacity $C(x_0, x_1)$ provided by the value-pair $\langle x_0, x_i\rangle$ can be defined by

$$C(\langle x_0, x_1\rangle) = C(\langle v, p\rangle) = h(p) - h(v) \times O_v, \tag{1}$$

where O_v is the number of bits required for recording the position of each pixel with gray value v and it is defined by $O_v = \lceil \log_2(\max(W, H))\rceil \times 2$.

Let P_m denote the set of m gray value-pairs for embedding the data and it is defined by $P_m = \{\langle x_{00}, x_{01}\rangle, \langle x_{10}, x_{11}\rangle, \ldots, \langle x_{(m-1)0}, x_{(m-1)1}\rangle\}$ where $x_{i0} < x_{i1}$; for $i \neq j$, the intersection of two intervals (x_{i0}, x_{i1}) and (x_{j0}, x_{j1}) is empty, i.e. $(x_{i0}, x_{i1}) \cap (x_{j0}, x_{j1}) = \phi$. The set P_m provides $C(P_m)$-bit capacity for embedding data where $C(P_m) = \sum_{i=0}^{m-1} C(\langle x_{i0}, x_{i1}\rangle)$. For the element $\langle x_{m0}, x_{m1}\rangle \in P_m$, adding the element $\langle x_{m0}, x_{m1}\rangle$ to the set P_m can increase $C(\langle x_{k0}, x_{k1}\rangle)$-bit capacity which is defined by

$$\Delta C(P_m, \langle x_{m0}, x_{m1}\rangle) = \begin{cases} C(\langle x_{m0}, x_{m1}\rangle), & \text{if } (x_{m0}, x_{m1}) \cap (x_{i0}, x_{i1}) = \phi \text{ for all } \{x_{i0}, x_{i1}\} \in P_m, \\ -\infty, & \text{otherwise.} \end{cases} \tag{2}$$

In Eq. (2), $\Delta C(P_m, \langle x_{m0}, x_{m1}\rangle) = -\infty$ means that $\langle x_{m0}, x_{m1}\rangle$ is illegal and prohibitive to be added to the set P_m for embedding the data.

According to the increasing function $\Delta C(P_m, \langle x_{m0}, x_{m1}\rangle)$, we now discuss how to select the most suitable gray value-pairs to maximize the embedding capacity of $H(I)$. Let $C^*(k)$ denote the maximal embedding capacity provided that the most suitable k gray value-pairs have been determined; let $C^*(k, \langle x_{j0}, x_{j1}\rangle)$ denote the maximal embedding capacity that we can embed the hidding data into the image I provided that the most suitable k gray value-pairs including the gray value-pair $\langle x_{j0}, x_{j1}\rangle$ have been determined and these k gray value-pairs are denoted by the set $P^*_{k,\langle x_{j0}, x_{j1}\rangle}$.

For $k = 1$, there are at most $32,640$ $\left(= \sum_{i=1}^{255} i\right)$ gray value-pairs to be considered and $C^*(1)$ is $i = 1$ given by

$$C^*(1) = \max_{P^*_{1,\langle x_{j0}, x_{j1}\rangle} \in P^*_{1,A}} (C^*(1, \langle x_{j0}, x_{j1}\rangle)), \tag{3}$$

where $C^*(1, \langle x_{j0}, x_{j1}\rangle) = C(\langle x_{j0}, x_{j1}\rangle)$, $P^*_{1,\langle x_{j0}, x_{j1}\rangle} = \langle x_{j0}, x_{j1}\rangle$, and $P^*_{1,A} = \cup_j P^*_{1,\langle x_{j0}, x_{j1}\rangle}$. For $k = 2$, we thus have:

$$C^*(2, \langle x_{j0}, x_{j1}\rangle) = \max_{P^*_{1,\langle x_{i0}, x_{i1}\rangle} \in P^*_{1,A}} (C^*(1, \langle x_{i0}, x_{i1}\rangle) + \Delta C(P^*_{1,\langle x_{i0}, x_{i1}\rangle}, \langle x_{j0}, x_{j1}\rangle)). \tag{4}$$

On the other hand, the set $P^*_{2,\langle x_{j0}, x_{j1}\rangle}$, can be determined by

$$P^*_{2,\langle x_{j0}, x_{j1}\rangle} = \langle x_{j0}, x_{j1}\rangle \cup \arg \max_{P^*_{1,\langle x_{i0}, x_{i1}\rangle} \in P^*_{1,A}} (C^*(1, \langle x_{i0}, x_{i1}\rangle) + \Delta C(P^*_{1,\langle x_{i0}, x_{i1}\rangle}, \langle x_{j0}, x_{j1}\rangle)). \tag{5}$$

From $C^*(2, \langle x_{j0}, x_{j1}\rangle)$ and $P^*_{2,\langle x_{j0}, x_{j1}\rangle}$, the value of $C^*(2)$ is obtained by

$$C^*(2) = \max_{P^*_{2,\langle x_{j0}, x_{j1}\rangle} \in P^*_{2,A}} (C^*(2, \langle x_{j0}, x_{j1}\rangle)), \tag{6}$$

where $P^*_{2,A} = \cup_j P^*_{2,\langle x_{j0}, x_{j1}\rangle}$. By the same arguments, for $3 < k < 128$, where '128' denotes the maximal number of the gray value-pairs that we can select for embedding data, $C^*(k)$ is defined by

$$C^*(k) = \max_{P^*_{k,\langle x_{j0}, x_{j1}\rangle} \in P^*_{k,A}} (C^*(k, \langle x_{j0}, x_{j1}\rangle)), \tag{7}$$

where

$$(C^*(k, \langle x_{j0}, x_{j1}\rangle)) = \max_{P^*_{k-1,\langle x_{i0}, x_{i1}\rangle} \in P^*_{k-1,A}} (C^*(k-1, \langle x_{i0}, x_{i1}\rangle) + \Delta C(P^*_{k-1,\langle x_{i0}, x_{i1}\rangle}, \langle x_{j0}, x_{j1}\rangle)),$$

$$P^*_{k,\langle x_{j0}, x_{j1}\rangle} = \langle x_{j0}, x_{j1}\rangle \cup \arg \max_{P^*_{k-1,\langle x_{i0}, x_{i1}\rangle} \in P^*_{k-1,A}} (C^*(k-1, \langle x_{i0}, x_{i1}\rangle) + \Delta C(P^*_{k-1,\langle x_{i0}, x_{i1}\rangle}, \langle x_{j0}, x_{j1}\rangle)),$$

and $P^*_{k,A} = \cup_j P^*_{k,\langle x_{j0}, x_{j1}\rangle}$. Consequently, the maximal embedding capacity of $H(I)$, $C^*_h(H(I))$, can be determined by

$$C^*_h(H(I)) = \max_{1 \leqslant k \leqslant 128} C^*(k). \tag{8}$$

3.2. The proposed dynamic programming-based capacity maximization algorithm

Following the two kernel equations derived in the last subsection, Eqs. (7) and (8), this subsection presents our proposed dynamic programming-based inversible data hiding algorithm to achieve the capacity maximization goal. For realizing the proposed dynamic programming-based approach, we use the array M with size $128 \times 256 \times 256$ to record all the values of $C^*(k, x_{j0}, x_{j1})$'s and we have $M[k-1][x_{j0}][x_{j1}] = C^*(k, \langle x_{j0}, x_{j1}\rangle)$; the array N with size $128 \times 256 \times 256$ is used to record all the found gray value-pairs which are corresponding to the optimal path determined at the end of the dynamic programming

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

process; the array P with size 1×128 is used to record the most suitable gray value-pairs in the array N. Our proposed dynamic programming-based algorithm consists of the following six steps:

Step 1. Initialize each entry of the three arrays M, N, and P to be $-\infty$, $\langle -1,-1\rangle$, and $\langle -1,-1\rangle$, respectively. From $C^*(1,\langle x_{j0},x_{j1}\rangle) = C(\langle x_{j0},x_{j1}\rangle)$ and $P^*_{1,\langle x_{j0},x_{j1}\rangle} = \langle x_{j0},x_{j1}\rangle$, we set $M[0][x_{j0}][x_{j1}] = C(\langle x_{j0},x_{j1}\rangle)$ and $N[0][x_{j0}][x_{j1}] = \langle x_{j0},x_{j1}\rangle$ for $0 \leqslant < x_{j0} \leqslant 254$, $1 < x_{j1} \leqslant 255$, and $x_{j0} < x_{j1}$. In addition, set $P^*_{1,A} = \cup_j P^*_{1,\langle x_{j0},x_{j1}\rangle} = \cup_j N[0][x_{j0}][x_{j1}]$ to save all gray value-pairs $\langle x_{j0},x_{j1}\rangle$'s. Set the variable k to be 2.

Step 2. Perform

$$M[k-1][x_{j0}][x_{j1}] = C^*(k,\langle x_{j0},x_{j1}\rangle) = \max_{P^*_{k-1,\langle x_{i0},x_{i1}\rangle}\in P^*_{k-1,A}} (C^*(k-1,\langle x_{i0},x_{i1}\rangle) + \Delta C(P^*_{k-1,\langle x_{i0},x_{i1}\rangle},\langle x_{j0},x_{j1}\rangle)),$$

and

$$N[k-1][x_{j0}][x_{j1}] = \arg\max_{P^*_{k-1,\langle x_{i0},x_{i1}\rangle}\in P^*_{k-1,A}} (C^*(k-1,x_{i0},x_{i1}) + \Delta C(P^*_{k-1,\langle x_{i0},x_{i1}\rangle},\langle x_{j0},x_{j1}\rangle)).$$

In addition, we set $P^*_{k,\langle x_{j0},x_{j1}\rangle} = \langle x_{j0},x_{j1}\rangle \cup N[k-1][x_{j0}][x_{j1}]$ and $P^*_{k,A} = \cup_j P^*_{k,\langle x_{j0},x_{j1}\rangle}$, and then by Eq. (7), the maximal embedding capacity $C^*(k)$ can be calculated.

Step 3. If $C^*(k) \neq -\infty$ or $k < 128$, perform $k = k + 1$ and go to Step 2; otherwise, go to Step 4.

Step 4. By Eq. (8), the maximal embedding capacity of the histogram, $C^*_h(H(I))$, is calculated by

$$C^*_h(H(I)) = \max_{1\leqslant k\leqslant 128} C^*(k)$$

and the number of gray value-pairs results in maximal embedding capacity determined by

$$k^* = \arg\max_{1\leqslant k\leqslant 128} C^*(k).$$

Further, the k^*th most suitable gray value-pair is recorded in $P[k^*-1]$ and it yields

$$P[k^*-1] = \arg\max_{\langle x_{j0},x_{j1}\rangle} M[k^*-1][x_{j0}][x_{j1}].$$

Set $\ell = k^* - 1$.

Step 5. Determine the ℓth most suitable peak-valley pair by performing

$$P[\ell-1] = N[\ell][x_0^*][\ell_1^*],$$

where $\langle x_0^*, x_1^*\rangle = P[\ell]$.

Step 6. Perform $\ell = \ell - 1$. If $\ell < 1$, we stop the algorithm; otherwise, go to Step 5.

After completing the above dynamic programming-based algorithm, the most suitable k^* gray value-pairs, i.e. the determined k^* peak-valley pairs, have been determined to maximize the embedding capacity. Based on each determined peak-valley pair, the embedding procedure mentioned in Section 2 is adopted to embed the proper substring of the hiding data into the original image. After examining all the determined peak-valley pair, the embedding procedure can iteratively embed all the hiding data with $C^*(k^*)$ bits into the original image.

4. Experimental results

In this section, first some experimental results are demonstrated to evaluate the embedding capacity comparison, the image quality, and the execution-time requirement between Ni et al.'s reversible data hiding algorithm and ours. Further, we compare our proposed algorithm with Chang et al.'s reversible data algorithm [4] in terms of the same three evaluation items. All the concerned three reversible data hiding algorithm are run on the IBM compatible Pentium IV microprocessor with 3.2 GHz and 1.5 GB RAM. The operating system is MS-Windows XP and the program developing environment is Borland C++ Builder 6.0.

Based on three testing map images as shown in Fig. 2a–c with sizes 739×814, 589×719, and 683×822, respectively, Table 1a indicates that the average embedding capacity gain is 23,932 bits and the average embedding capacity improvement ratio is 9% under the overhead listed in Table 1b. In fact, our proposed dynamic programming-based approach reveals that this average embedding capacity gain is the maximal embedding capacity gain that we can make for the three testing images. As shown in Table 1c, our proposed algorithm has little image quality degradation. As shown in Table 1d, Ni et al.'s algorithm is much faster than our proposed algorithm. Table 1e indicates that surprisingly, our proposed dynamic programming-based algorithm only needs 15 peak-valley pairs in average such that the maximal embedding capacity can be achieved while Ni et al.'s algorithm needs 40 peak-valley pairs in average.

Based on the same three testing images, we also compare Chang et al.'s reversible data hiding algorithm with our proposed algorithm. Table 2 indicates that in average, our proposed algorithm has much better image quality and execution-time performance although Chang et al.'s algorithm has better embedding capacity.

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

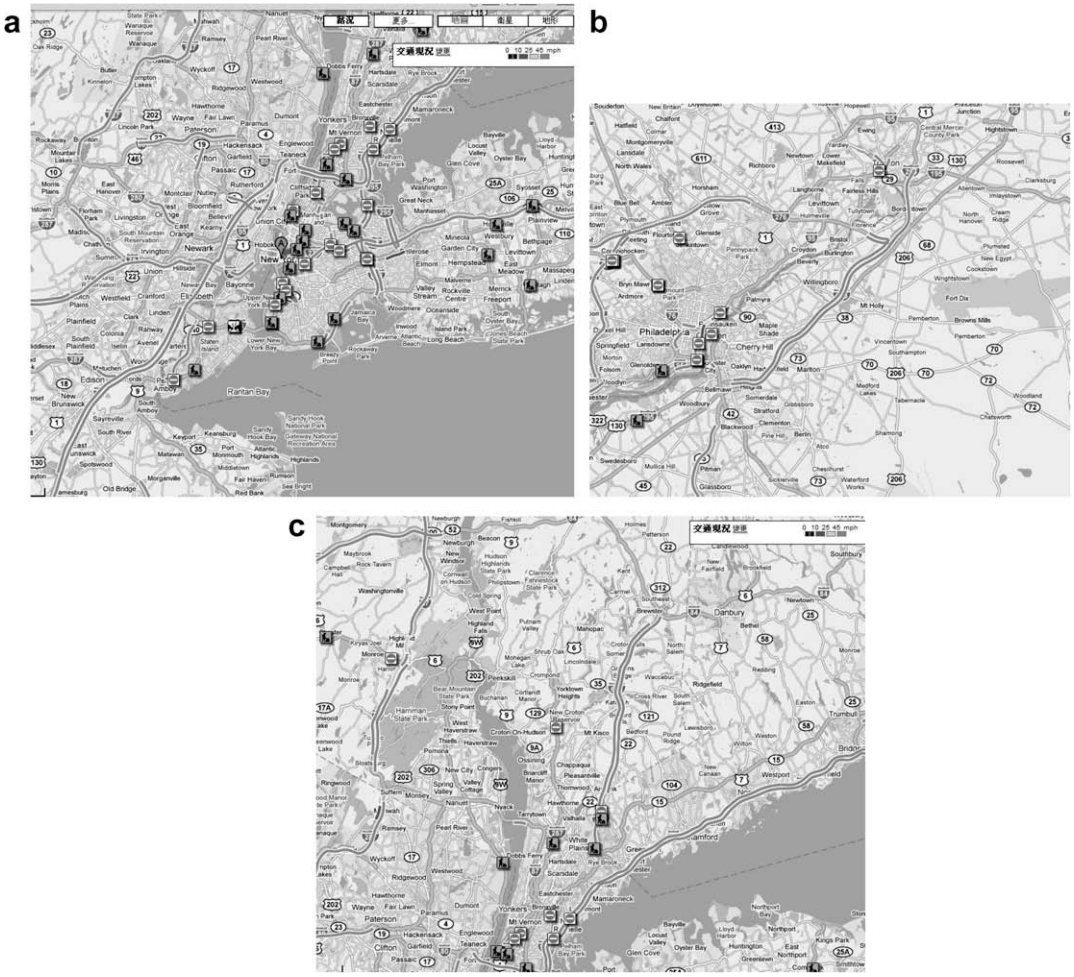

Fig. 2. Three testing images.

Table 1
Performance comparison between Ni et al.'s algorithm and our proposed algorithm.

	Fig. 2a	Fig. 2b	Fig. 2c
(a) *Embedding capacity comparison*			
Ni et al.'s algorithm	269,655	228,324	279,719
The proposed algorithm	309,251	242,418	297,826
Embedding capacity gain	39,596	14,094	18,107
Average improvement ratio	9%		
(b) *Overhead comparison*			
Ni et al.'s algorithm	56,160	41,320	57,500
The proposed algorithm	43,700	31,080	49,460
(c) *PSNR comparison*			
Ni et al.'s algorithm	49.0	49.6	49.2
The proposed algorithm	48.7	49.0	48.9
(d) *Execution-time comparison*			
Ni et al.'s algorithm	0.0009	0.001	0.0009
The proposed algorithm	11	23	21
(e) *Number of peak-valley pairs selected*			
Ni et al.'s algorithm	45	36	40
The proposed algorithm	9	24	13

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

Table 2
Performance comparison between Chang et al.'s algorithm and our proposed algorithm.

	Chang et al.' s algorithm	The proposed algorithm
Embedding capacity	326,139	283,165
PSNR	34.5	48.2
Execution-time	41	15

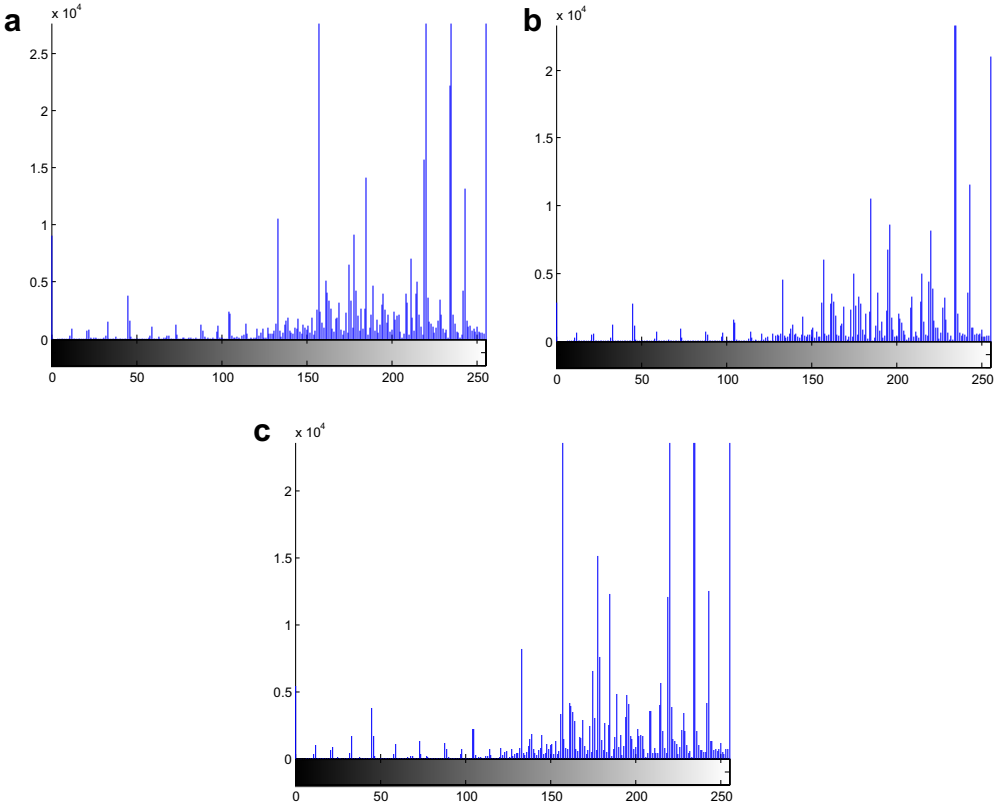

Fig. 3. Histograms of Fig. 1. (a) Histogram of Fig. 1a. (b) Histogram of Fig. 1b. (c) Histogram of Fig. 1c.

Table 3
Embedding capacity comparison between Ni et al.'s algorithm and our proposed algorithm for Lena image, pepper image, and baboon image.

	Lena	Pepper	Baboon
Ni et al.'s algorithm	5590	5339	5877
The proposed algorithm	5590	5356	5895

Although our proposed algorithm provides 9% average embedding capacity improvement ratio for map images, the histogram of each natural image usually generates few peak-valley pairs to embed the hiding data and it limits the embedding capacity. Fig. 3 illustrates the three histograms of the three map images as shown in Fig. 2. Fig. 4 shows the three histograms of the three natural images, Lena, pepper, and baboon. When comparing Figs. 3 and 4, it can be observed that the three map images as shown in Fig. 2 contain more peak-valley pairs than those in the three natural images. Table 3 shows the embedding capacity comparison between Ni et al.'s algorithm and our proposed algorithm for Lena image, pepper image, and baboon image, and we find that the embedding capacity of Ni et al.'s algorithm is very close to that of our proposed algorithm.

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

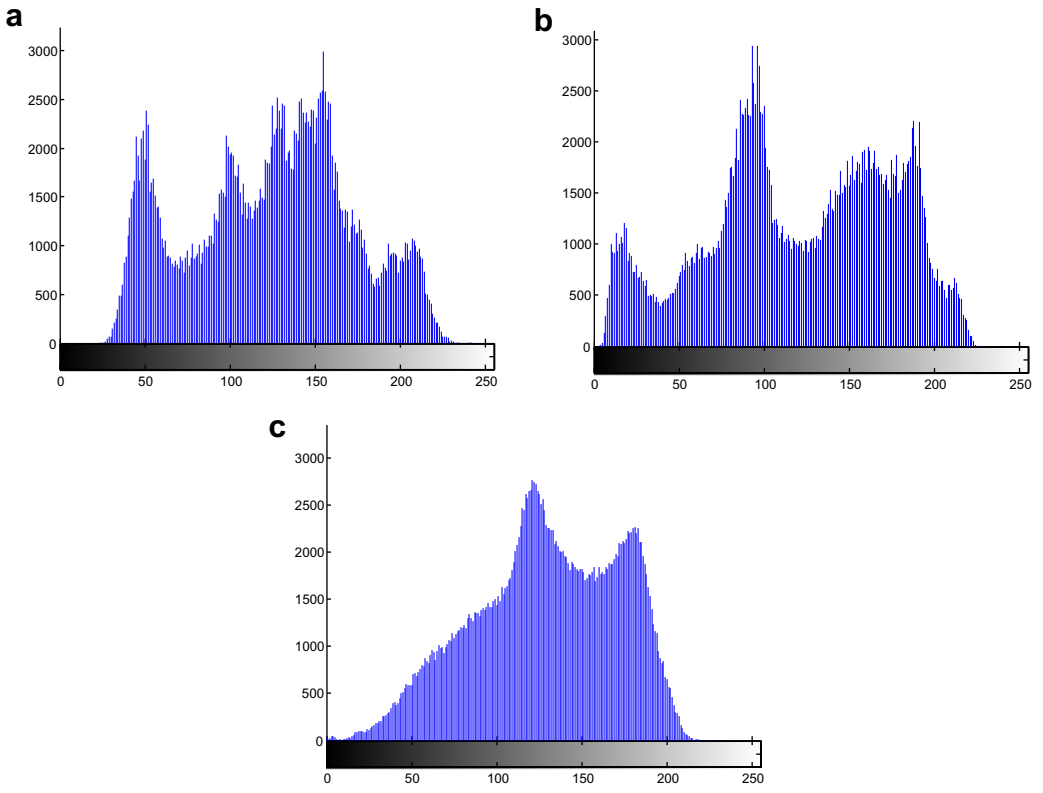

Fig. 4. The histogram of three images, Lena, pepper, and baboon. (a) Histogram of Lena. (b) Histogram of pepper. (c) Histogram of baboon.

5. Conclusion

Instead of using the greedy approach by Ni et al.'s algorithm, our proposed dynamic programming-based reversible data hiding algorithm has been presented. The proposed algorithm can determine the most suitable peak-valley pairs such that the embedding capacity can be maximized and the high image quality can be kept. However, the proposed algorithm has some execution-time degradation. The major contribution of this paper is that the proposed dynamic programming-based algorithm can maximize the embedding capacity of Ni et al.'s algorithm. Furthermore, the comparison between our proposed dynamic programming-based algorithm and the reversible data hiding algorithm by Chang et al. is investigated.

According to our experiments, although our proposed algorithm has good performance for map images, the histogram of each natural images usually generates few peak-valley pairs to embed the hiding data and it limits the embedding capacity. How to improve Ni et al.'s algorithm and apply it to natural images is an interesting research issue. Further, how to reduce the number of bits required for recording the position of each pixel corresponding to the valley point is another interesting research issue.

References

[1] M.U. Celik, G. Sharma, A.M. Tekalp, E. Saber, Reversible data hiding, in: Proceedings of the IEEE International Conference on Image Process, vol. 2, September, 2002, pp. 157–160.
[2] C.C. Chang, G.M. Chen, M.H. Lin, Information hiding based on search-order coding for VQ indices, Pattern Recognition Letters 25 (11) (2004) 1253–1261.
[3] C.C. Chang, P.Y. Tsai, M.H. Lin, SVD-based digital image watermarking scheme, Pattern Recognition Letters 26 (10) (2005) 1577–1586.
[4] C.C. Chang, C.C. Lin, Y.H. Chen, A reversible data embedding scheme using differences between original and predicted pixel values, IET Information Security 2 (2) (2008) 35–46.
[5] T.J. Chuang, J.C. Lin, W.H. Tsai, A new efficient approach to image hiding by digit number transformation, Pattern Recognition and Image Analysis 10 (3) (2000) 309–314.
[6] K.L. Chung, C.H. Shen, L.C. Chang, A novel SVD- and VQ-based image hiding scheme, Pattern Recognition Letters 22 (9) (2001) 1051–1058.
[7] K.L. Chung, W.N. Yang, Y.H. Huang, S.T. Wu, Y.C. Hsu, On SVD-based watermarking algorithm, Applied Mathematics and Computation 188 (1) (2007) 54–57.
[8] I.J. Cox, M. Miller, J. Bloom, Digital Watermarking, Morgan Kaufmann, San Francisco, CA, 2001.
[9] J. Cox, J. Kilian, T. Leighton, T. Shamoon, Secure spread spectrum watermarking for multimedia, IEEE Transactions on Image Processing 6 (12) (1997) 1673–1687.
[10] J. Fridrich, M. Goljan, R. Du, Invertible authentication, in: Proceedings of the SPIE, Security and Watermarking of Multimedia Contents, San Jose, CA, January 2001, pp. 197–208.

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066

[11] M. Gkizeli, D.A. Pados, M.J. Medley, Optimal signature design for spread-spectrum steganography, IEEE Transactions on Circuits and Systems for Video Technology 16 (2) (2007) 391–405.
[12] M. Goljan, J. Fridrich, R. Du, Distortion-free data embedding for images, in: Proceedings of the Fourth International Hiding Workshop, Pittsburgh, PA, April 2001, pp. 27–41.
[13] C.W. Honsinger, P. Jones, M. Rabbani, J.C. Stoffel, Lossless recovery of an original image containing embedded data, US Patent 6,278,791, 2001.
[14] J. Irvine, D. Harle, Data Communications and Networks: An Engineering Approach, Wiley, New York, 2002.
[15] M. Jo, H.D. Kim, A digital image watermarking scheme based on vector quantization, IEICE Transactions on Information and System E85-D (6) (2002) 1054–1056.
[16] S.L. Li, K.C. Leung, L.M. Cheng, C.K. Chan, Data hiding in images by adaptive LSB substitution based on the pixel-value differencing, Innovative Computing, Information and Control 3 (2006) 58–61.
[17] R. Liu, T. Tan, An SVD-based watermarking scheme for protecting rightful ownership, IEEE Transactions on Multimedia 4 (1) (2002) 121–128.
[18] C.S. Lu, S.K. Huang, C.J. Sze, H.Y. Mark Liao, Cocktail watermarking for digital image protection, IEEE Transactions on Multimedia 2 (4) (2000) 209–224.
[19] C.S. Lu, J.R. Chen, K.C. Fan, Real-time frame-dependent video watermarking in VLC domain, Signal Processing: Image Communication 20 (7) (2005) 624–642.
[20] B. Macq, F. Deweyand, Trusted headers for medical images, in: DFG VIII-DII Watermarking Workshop, Erlangen, Germany, October 1999.
[21] L.M. Marvel, C.G. Boncelet, C.T. Retter, Spread spectrum image steganography, IEEE Transactions on Image Processing 8 (8) (1999) 1075–1083.
[22] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Transactions on Circuits and Systems for Video Technology 16 (3) (2006) 354–362.
[23] A.Z. Tirkel, C.F. Osborne, R.G. Schyndel, Image watermarking—a spread spectrum application, in: Proceedings of the IEEE Fourth International Symposium on Spread Spectrum Techn. Applicat., vol. 2, December 1997, pp. 785–789.
[24] D.C. Wu, W.H. Tsai, Spatial-domain image hiding using image differencing, IEE Proceedings – Vision, Image and Signal Processing 147 (1) (2000) 29–37.
[25] G. Xuan, J. Zhu, J. Chen, Y.Q. Shi, Z. Ni, W. Su, Distortionless data hiding based on integer wavelet transform, IEE Electronic Letters 38 (25) (2002) 1646–1648.

Please cite this article in press as: K.-L. Chung et al., Capacity maximization for reversible data hiding based on dynamic ..., Appl. Math. Comput. (2008), doi:10.1016/j.amc.2008.10.066