A High Capacity Reversible Watermarking Scheme

Marc Chaumont[a,b] and William Puech[a,b]

[a] Laboratory LIRMM, UMR CNRS 5506, University of Montpellier II,
161, rue Ada, 34392 Montpellier cedex 05, France.

[b] University of Nîmes, Place Gabriel Péri, 30000 Nîmes, France.

ABSTRACT

Many **reversible** watermarking solutions have been proposed since 1996: spread-spectrum approaches, circular interpretation on histogram approaches, lossless compression approaches, expansion approaches and histogram approaches. In this paper, we propose a solution whose embedding capacity goes beyond all those reversible schemes. For certain images, the reach payload of the proposed method is over **2 bpp**. This solution is an improvement of the Coltuc reversible watermarking approach published in ICIP'2007.

Keywords: Reversible watermarking, high-capacity, congruence-based watermarking.

1. INTRODUCTION

Reversible watermarking methods for images own three major requirements: a high embedding payload, a small complexity and a good visual quality. In this paper we mostly look at a high payload scheme with a small complexity. Our algorithm does not care about the quality criteria. It produce a kind of "salt-and-pepper" noise.

> "Although salt-and-pepper artifacts might appear ugly, it must be remembered that they will be removed when the original Work is recovered. ... For example, suppose the watermarked Works are used only for browsing. ... In such a scenario, the watermarked Works need only to be recognizable, and salt-and-pepper artifacts might not be a serious problem. (Digital Watermarking and Steganography, 2007, p. 382)[1]".

We then believe that this approach might been used for medical or satellite images. Moreover, it has already been successfully used for color information protection in.[2]

Since 1996, lots of reversible watermarking solutions have been proposed: spread-spectrum approaches,[3] circular interpretation on histogram approaches,[4] lossless compression approaches,[5,6] expansion approaches[7] and histogram approaches.[8,9] Our proposed solution allows to obtain an embedding-payload going beyond all the previous reversible schemes. Moreover, the algorithm complexity is very small. The proposed solution in this paper is an improvement of the Coltuc reversible watermarking approach.[10] More precisely, our objective is to correct the Coltuc scheme. Indeed, in some cases the process of[10] can not be reverted due to a problem of dependencies during the decoding process. We have then re-formulated its scheme in order to clarified it. We also provide lots of experimental results. In Section 2 we deal with the new watermarking scheme. In Section 3 we treat more precisely few technical points and have a discussion about the scheme. Finally, in Section 4 and 5, we give results and conclude.

Send correspondence to Marc.Chaumont@lirmm.fr.

2. ALGORITHM PRINCIPLE

In the same way as the method proposed by,[10] our algorithm lies on congruence computations. But in,[10] Coltuc proposes only two possible states for each pixel of an image. In our approach, we define **three possible states** for a pixel:

- the state ***embedding*** which corresponds to a pixel *embedding* an integer coefficient belonging to $[1, n]$,
- the state ***to-correct*** which corresponds to a pixel that has been modified but *does not embed* any information; this pixel will be *corrected* during the reverting process,
- the state ***original*** which corresponds to an *original* pixel (i.e unchanged). This state is new compared to those defined in.[10]

Lets define a constant integer value n greater or equals to 3. Lets also define the T transform which takes two integers x_1 and x_2 as input and return an integer:

$$\begin{aligned} T &: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \\ T(x_1, x_2) &= (n+1).x_1 - n.x_2. \end{aligned}$$

In order to clarify the paper, we will name the coding process, the act of embedding a message into an image and the decoding process, the act of extracting the message and rebuilding the original image. Lets now define the three possible states and the coding-decoding algorithms.

2.1 Embedding state

A pixel i in the *embedding* state is a pixel such that:

$$0 \leq T(I(i), I(i+1)) \ \ and \ \ T(I(i), I(i+1)) + n \leq L, \tag{1}$$

with I the original image (of N pixel size) whose grey-levels belongs to $[0, L]$. In the case of 8-bits images, L equals to 255. All pixels i which are in the *embedding* state:

1. are T transformed such that $I_T(i) = T(I(i), I(i+1))$, with $I_T(i)$ the resulting transformed pixel,
2. **must** then embed **a** coefficient w **belonging to** $[1, n]$ such that $I_w(i) = I_T(i) + w$.

Note that after the embedding of a coefficient w belonging to $[1, n]$, it is impossible to recover $I(i)$ with the only knowledge of $I(i+1)$. Indeed $I_w(i) = (n+1).I(i) - n.I(i+1) + w$ with $w \neq 0$, thus $I(i) = \frac{I_w(i)+n.I(i+1)-w}{n+1}$ and so:

$$(I_w(i) + n.I(i+1)) \ mod \ (n+1) \neq 0. \tag{2}$$

This **congruence property** (2) allows to detect an *embedding* pixel during the decoding process. Note that during the decoding process, the $I_w(i+1)$ pixel should have been previously reverted to $I(i+1)$ in order to compute this congruence. This implies that the image order scan used during the decoding process is the opposite of the one used during the coding process.

2.2 To-correct state

A pixel i in the *to-correct* state is a pixel such that the negation of equation (1) is true:

$$T(I(i), I(i+1)) < 0 \ \ or \ \ T(I(i), I(i+1)) + n > L.$$

All pixels that are in this *to-correct* state are then modified such that:

$$\begin{aligned} & c \leftarrow (I(i) + n.I(i+1)) \ mod \ (n+1); \\ & if \ (I(i) - c) < 0 \ \ then \ \ c \leftarrow -(n+1-c); \\ & I_w(i) \leftarrow I(i) - c. \end{aligned} \tag{3}$$

The c coefficients belong to $[-n, n]$* and are embedded (into the *embedding* pixels) in order to enable the reversibility of the *to-correct* pixels during the decoding process. We name the c coefficients the **corrective codes**. Note that after the modification expressed by equation (3), pixel $I_w(i)$ checks the property:

$$(I_w(i) + n.I(i+1)) \ mod \ (n+1) = 0. \tag{4}$$

This **congruence property** (4) allows to detect a *to-correct* pixel at the reverting process. Note that at the decoding process the $I_w(i+1)$ pixels should have been previously reverted to $I(i+1)$ in order to compute this congruence.

2.3 Original state

Given an image order scan for the coding, a pixel in the *original* state (i.e. unmodified pixel) must always be present just before a pixel in the *to-correct* state. For a top-bottom-left-right scan order, if a pixel i is in the *to-correct* state, then the pixel $i-1$ **must be** in the *original* state. In order to ensure this strong property (*original* and *to-correct* pixels go by pairs), during the scan, when a pixel i is detected as a *to-correct* one, a forward research is proceeded in order to find the next *embedding* one (noted *next*). *Original* and *to-correct* states are then alternates between the i (or $i-1$) position and the $next - 1$ position. See 3.1 for more details and Appendix section for the algorithms.

This grouping constraint (additional state to Coltuc scheme[10]†) breaks the problematic dependencies during the decoding process. Remember that during the decoding, the image scan order is inverted. A *to-correct* pixel i may not be reverted immediately if its associated corrective code is still not extracted. Nevertheless, because the pixel $i-1$ is an *original* pixel, the pixel $i-2$ may be treated immediately and the decoding process may continue (pixel i will be corrected later, in a second pass).

2.4 Coding and decoding algorithms

The **coding** process is composed of two steps (see Appendix section for the coding algorithm):

1. classify each pixel in one of the three states: *embedding*, *to-correct*, *original*,
2. embed into the *embedding* pixels, the watermark made of corrective codes plus the message.

Note that the image scan order has been chosen to be from top to bottom and from left to right.

For the **decoding** process, the image scan order is inverted; it is perform from bottom to top and from right to left. The decoding process is also composed of two steps (see Appendix section for the decoding algorithm):

1. extract the watermark from the *embedding* pixels, revert (during the scan) all those pixels and localize the *to-correct* pixels,
2. from the extracted watermark retrieve the corrective codes and the message, and correct the *to-correct* pixels.

3. FEW TECHNICAL DETAILS

3.1 About pairs of to-correct and original pixels

During the first step of the coding process the image is scan from top to bottom and from left to right in order to classify each pixel in one of the three states: *embedding*, *to-correct* and *original*. A pixel is classified in the *embedding* state if equation (1) is verified. When equation (1) is not verified, one have to alternate *original* and *to-correct* sites until reaching an *embedding* site. In this case, **we have** then to respect the constraint of producing **pairs** of *original* and *to-correct* sites.

Lets note i the position of the encountered pixel which does not verify equation (1). Lets note j the position of the first encountered pixel starting from i which verifies equation (1). If $j - i$ is odd (respectively even) then alternate *original* and *to-correct* sites from $i-1$ (respectively i) to $j-1$ (respectively $j-1$).

*In,[10] the author forget the case $I(i) - c < 0$; this mistake implies a wrong range $c \in [0, n]$.

†Without this additional constraint, the scheme proposed in[10] does not work in lots of cases.

3.2 About the embedding of corrective codes

During the first step of the **coding process**, the image is scanned from top to bottom and from left to right, and when a *to-correct* site is found, a corrective code is built. At the end of the first step of the coding, the coder own a list of corrective codes which are arranged in the find order. This list is then concatenated to the message before embedding. **During the decoding process**, after the first step, the corrective codes list is extracted and *to-correct* pixels are then corrected.

Another important point is that corrective codes belong to $[-n, n]$ and embedding is possible only with coefficients belonging to $[1, n]$. One can entropically encode (Huffman or Arithmetic coding) the corrective codes and then embed the obtained binary vector with respect to the embedding sites payload (i.e $log_2(n)$ bits per pixel (bpp)). In a first approach and for simplicity we do not have chosen this solution. Our solution consists in embedding:

- either directly the corrective code **plus one**, if the corrective code belongs to [0, n-3];
- either two coefficients w_1 and w_2, if the corrective code c belongs to $[-n, -1]$ or $[n - 2, n]$. The first coefficient w_1 is a special code used to communicate to the decoder. If $w_1 = n - 1$, this signify to the decoder that w_2 represents a corrective code belonging to $[n - 2, n]$. If $w_1 = n$, this signify to the decoder that w_2 represents a corrective code belonging to $[-n, -1]$. Thus, the coefficient w_2 allows to recover the corrective code.

Equations below resume the method used to code a *corrective code* c belonging to $[-n, n]$ in order to obtain coefficient(s) belonging to $[1, n]$:

$$\begin{cases} \text{if } c \in [0, n-3] & \text{embed } w_1 = c + 1 & \\ \text{if } c \in [n-2, n] & \text{embed } w_1 = n - 1 & \text{and } w_2 = c - n + 3 \\ \text{if } c \in [-n, -1] & \text{embed } w_1 = n & \text{and } w_2 = -c \end{cases}$$

At the decoding step, the decoder extract a coefficient w_1 and, depending of the case, extract coefficient w_2:

$$\begin{cases} \text{if } w_1 \in [1, n-2] & \text{decode } c = w_1 - 1 \\ \text{if } w_1 = n - 1 & \text{decode } c = w_2 + n - 3 \\ \text{if } w_1 = n & \text{decode } c = -w_2 \end{cases}$$

3.3 Hiding payload

Lets k be the number of *embedding* pixels. Each *embedding* pixel allows the insertion of **one** coefficient belonging to the range $[1, n]$ which corresponds to an embedding-payload of $log_2(n)$ bits. The $N - k$ remaining pixels[‡] (*to-correct* and *original* pixels) represent $(N - k)/2$ **corrective codes**. Indeed, the *to-correct* and *original* pixels go by pairs and there is corrective codes only for the *to-correct* pixels. A **corrective code** belongs to the range $[-n, n]$ and is thus corresponding to a rate of $log_2(2n+1)$ bits. The theoretical **real** payload of the watermarking scheme equals to the embedding payload **minus** the **corrective codes** bitrate[§] i.e:

$$\frac{k}{N} log_2(n) - \frac{N-k}{2N} log_2(2n+1) \textit{ bits per pixel.} \tag{5}$$

When the number k of *embedding* pixels is close to N the payload is approximately of $log_2(n)$ bpp. For example, if $k \approx N$, the payload can be close to 2 bpp with $n = 4$.

[‡]N is the image size.

[§]In[10] Coltuc gives a theoretical payload of $\frac{k}{N} log_2(n) - \frac{N-k}{N} log_2(n+1)$ bpp which is in practical impossible.

3.4 Discussions

In the Coltuc scheme[10] the decoding process necessitates for each watermarked pixel $I_w(i)$ the knowledge of the original pixel $I(i+1)$. There is thus a dependency between each pixel going from the last one to the first one. Unfortunately, only two states are used in its scheme (named in our paper the *to-correct* state and the *embedding* state). During the decoding process (i.e message extraction + reversibility), for any watermarked pixel $I_w(i)$ all the pixels $I_w(i+1), I_w(i+2), ..., I_w(N)$ should have been recovered. This strong dependency yield to a **not always possible scheme**.

Indeed, in its scheme, some pixels are modified by adding them a coefficient (named in our paper **corrective code**). Those corrective codes must be embedded additionally to the user message. At the decoding process, if a pixel $I_w(i)$ has to be corrected and if the associated corrective code has not still been extracted, then the decoding process is impossible and the message extraction fails. For example, take the two last coefficients $I_w(N-1)$ and $I(N)$ and suppose that the pixel $I_w(N-1)$ should be corrected thanks to a corrective code. This correcting code is supposed to be embedded in the image but is still not extracted. It is thus impossible to revert the pixel $I_w(N-1)$ and thus impossible to continue the decoding process. Other cases of impossibility may occur during the decoding process.

In order to break this dependency problem, we introduce the supplementary *original* state. *Original* pixels and *to-correct* pixels go by couple and are alternate if a suite of non-embedding pixels occurs. Those improvement enable to have a valid scheme and to proceed to multiple watermarking.

4. RESULTS

In this section, few results are given for classical grey-level images, 512×512, 8 bits per pixel. Table 1 gives the payload results with parameter **n=4**. The obtained payload is really impressive. For example, for the Lena image the embedding payload is 457 256 bits. To the author knowledge, the best reachable payload is with an expansion approach proposed in.[7] In,[7] Tian obtains a payload of 1.97 bpp $\equiv$ 516 794 bits for Lena image; In our proposed approach we reach a maximum payload of 2.06 bpp $\equiv$ 541 464 bits with **n=9**. We could moreover remark that the scheme own a very small computational complexity and that the payload may still be improved thanks to the corrective codes entropic coding.

Image	payload		PSNR
Airplane	457 256 bits	1.74 bpp	20.90 dB
Lena	445 294 bits	1.70 bpp	19.60 dB
Godhill	435 040 bits	1.66 bpp	18.56 dB
Peppers	418 710 bits	1.60 bpp	19.56 dB
Boat	409 444 bits	1.56 bpp	19.52 dB
Barbara	285 620 bits	1.09 bpp	18.82 dB
Baboon	195 608 bits	0.75 bpp	16.10 dB

Table 1. Payload on few classical grey-level images, 512×512, 8-bits per pixel with $\mathbf{n} = \mathbf{4}$.

In theory the algorithm may be run several times. In practice, with $\mathbf{n} = \mathbf{4}$, the algorithm succeed in embedding bits in a second algorithm execution only with Airplane and Lena images and the payload is very low (0.4 bpp for Lena and 0.03 bpp for Airplane). With Godhill, Peppers, Boat, Barbara and Baboon there were not enough payload for a second pass embedding. From equation (5), the first term is smaller than the second one.

As a consequence, for an higher embedding payload, it is most of the time more interesting to use the algorithm just with one pass execution and with a value for parameter **n** higher than 4. Indeed, as it could be observed in Figure 1, excepted for Baboon[¶] image, a higher payload is reached when $n > 4$.

Figures 2 and 3 give results of the embedding of a pseudo-random binary vector with parameter n=4 on the grey-level image Lena 512×512 shown in Figure 2(a) and on the grey-level image *head* 228×256 shown in Figure 3(a). Figure 2(b) and 3(b) show the resulting watermarked image. Figure 2(c) and 3(c) show the three

[¶]In general, the payload is week for strongly textured images and one should choose a small value for n.

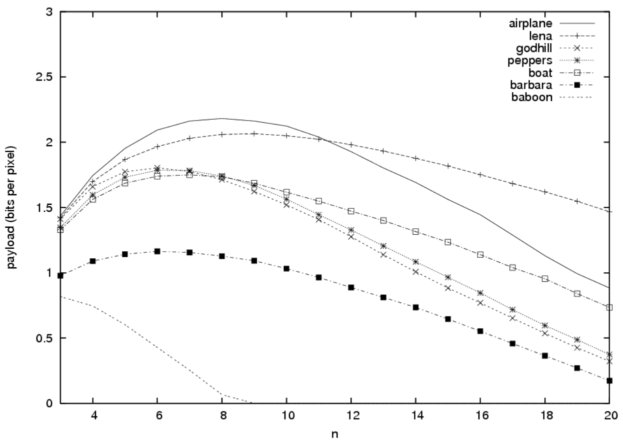
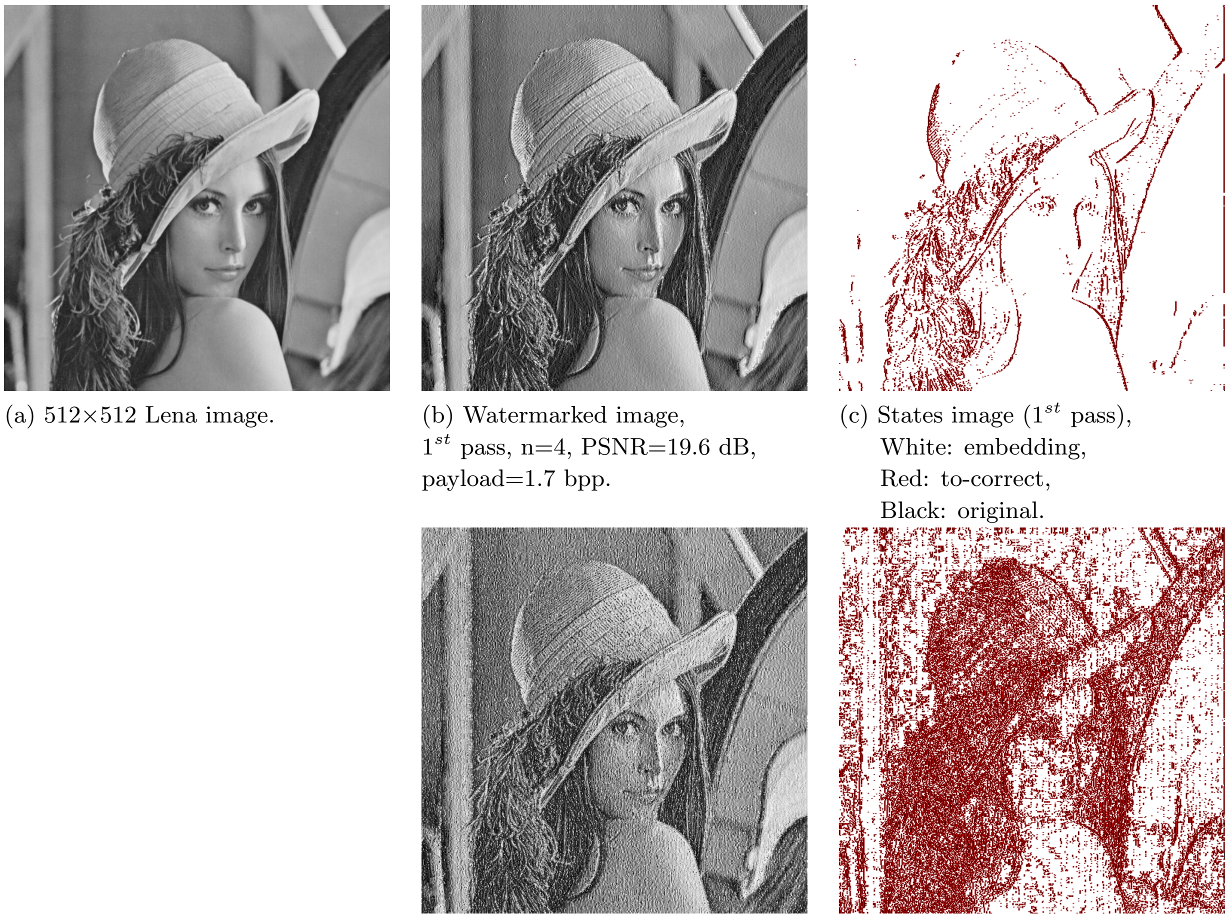


Figure 1. Real payload variation in function of the parameter **n** for few classical 512 × 512, 8 bits images.



(a) 512×512 Lena image.

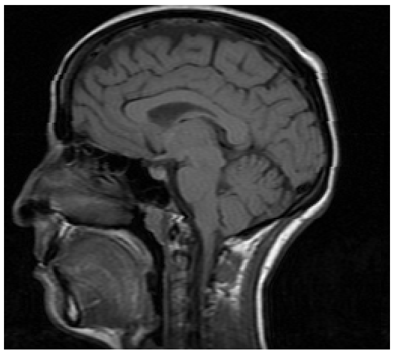
(b) Watermarked image, 1st pass, n=4, PSNR=19.6 dB, payload=1.7 bpp.

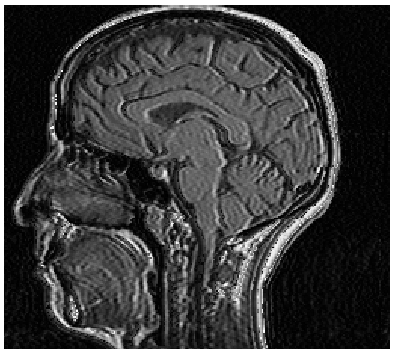
(c) States image (1st pass), White: embedding, Red: to-correct, Black: original.

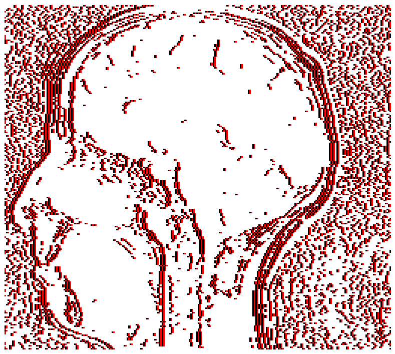
(d) Watermarked image, 2nd pass, n=4, PSNR=14 dB, combined payload=1.7+0.4 bpp.

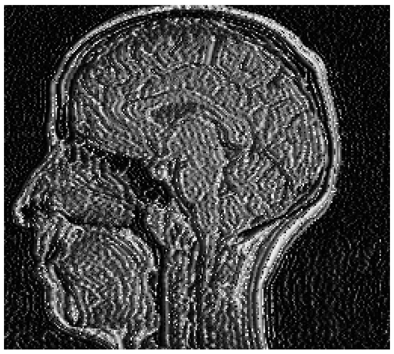
(e) States image (2nd pass).

Figure 2. Illustration of the algorithm on Lena 512 × 512 image, with n=4 and 2 passes. The total embedding **real** payload equals to **2.1 bpp** i.e **550 940 bits**.


(a) 228×256 *head* image.


(b) Watermarked image, 1^{st} pass, n=4, PSNR=19.8 dB, payload=0.99 bpp.


(c) States image (1^{st} pass), White: embedding, Red: to-correct, Black: original.


(d) Watermarked image, 2^{nd} pass, n=4, PSNR=14.3 dB, combined payload=0.99+0.04 bpp.

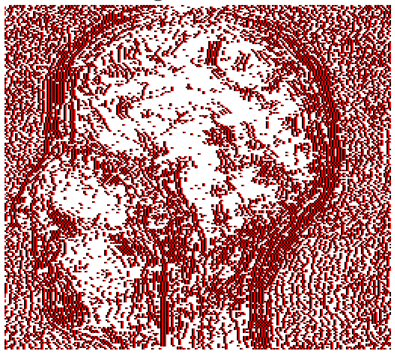

(e) States image (2^{nd} pass).

Figure 3. Illustration of the algorithm on *head* 228×256 medical image with n=4 and 2 passes. The total embedding **real** payload equals to **1.03 bpp** i.e **60 478 bits**.

different sites: in white, there are the pixels embedding a coefficient belonging to $[1, n]$, in red there are the pixels to-correct (those pixels produce a corrective code belonging to [-n, n]) and in black there are the non-modified original pixels. The watermarked image obtained after two consecutive embedding is shown in Figure 2(d) and 3(d). One can remark, on Figure 2(e) and 3(e), that in the second algorithm execution, there is lots of non-embedding sites (*to-correct* and *original* pixels).

5. CONCLUSION

To conclude, the proposed scheme in this paper improves the previous Coltuc scheme[10] by breaking dependencies and thus ensuring that watermarking may always be reverted. Moreover, those dependencies broken allows multiple consecutive watermarkings. Thus, embedding payload reach amazing values **around 2 bbp**. Such high capacities were never previously reached (see circular interpretation on histogram approaches,[4] lossless compression approaches,[5,6] expansion approaches,[7] and histogram approaches[8,9]). Future work should deal with the tradeoff between distortion and payload; indeed the current work always reach the maximum payload without taking into account distortion. Future work should also deal with the corrective codes compression and the used of a secret-key (modifying the order-scan) for the scheme' security.

ACKNOWLEDGMENTS

This investigation was supported in part by the TSAR project which is a french national project ANR-05-SSIA-0017-05 of the ANR ARA SSIA (*Agence Nationale de la Recherche, Action de Recherche Amont, Sécurité Systèmes Embarqués et Intelligence Ambiante*) and in part by the VOODDO project which is a french national project of the ANR Contenu et Interaction. We would also like to thank the Languedoc-Roussillon Region.

APPENDIX

Listing 1. *Embedding algorithm*

```
const Integer n;                                              // n ≥ 3

Procedure embed(Image: $I$, Message: $m$):$I_w$                // m coefficients $\in [0, n-1]$
begin

  List embedding;                                             // Declaration of the embedding list
  List codes;                                                 // Declaration of the corrective codes list

  // FIRST STEP
  for $i$ from 1 to $N-1$
  begin

    $t \leftarrow (n+1).I(i) - n.I(i+1)$;                       // T transform

    if (($0 \leq t$) and ($t+n \leq L$))
    then
      $I_w(i) \leftarrow t$;                                    // Apply T transformation to the i$^{th}$ pixel
      $embedding \leftarrow embedding \oplus i$;                // Add pixel i to the embedding list
    else
      $next \leftarrow$ look_forward_for_next_embedding_pixel($i$);   // $next$ is the next embedding pixel

      if ($next - i$ is odd)
      then
        $k \leftarrow i-1$;
        $I_w(i-1) \leftarrow I(i-1)$;                             // Pixel i-1 come back to its original value
        embedding $\leftarrow$ embedding $\ominus$ $i$;             // Remove pixel i from the embedding list
      else
        $k \leftarrow i$;
      end-if

      for $j$ from $k$ to $next-1$ alternate original and to_correct pixels
      begin
        if (it is to_correct turn)
        then
          $c \leftarrow (I(j) + n.I(j+1))$ $mod$ $(n+1)$;       // Positive code
          if (($I(j) - c) < 0$) then $c \leftarrow -(n+1-c)$;   // Negative code
          $I_w(j) \leftarrow I(j) - c$;
          $codes \leftarrow codes \oplus c$;                    // Add corrective code $c$ to the codes list
        end-if
      end-for
      $i \leftarrow next - 1$;                                  // Next iteration, goes to the $next$ embedding pixel
    end-if
  end-for

  Generate a coefficient sequence $w$ with the corrective code list $codes$ and the message $m$;

  // SECOND STEP : embedding OF THE WATERMARK $w$
  $\forall i \in embedding$  $I_w(i) \leftarrow I_w(i) + w(i)$;          // $w(i) \in [1, n]$
end
```

Listing 2. *Extraction algorithm*

```
Procedure extract(Image: $I_w$):$I$, $m$
begin

  List to_correct;                     // Declaration of the to_correct list

  $I \leftarrow I_w$;                  // copy $I_w$ into $I$

  // FIRST STEP: EXTRACTION OF THE WATERMARK $w$
  for i from N−1 to 1
  begin

    $v \leftarrow (I_w(i) + n.I(i+1)) mod (n+1)$;

    if ($v = 0$)                       // Congruence property test
    then                               // Case: pixel i is a to_correct pixel
      to_correct ← to_correct ⊕ $i$;   // Add pixel $i$ to the to_correct list
      $i \leftarrow i - 1$;            // Important lign: pixel $i-1$ is original and is not to treat
    else                               // Case: pixel i is an embedding pixel
      $w \leftarrow w \oplus v$;       // Concatenate coefficient $v$ to the watermark $w$
      $I_w(i) \leftarrow I_w(i) - v$;  // Remove coefficient $v$
      $I(i) \leftarrow \frac{I_w(i)+n.I(i+1)}{n+1}$   // Invert T transformation
    end−if
  end

  Retrieve from the coefficient sequence $w$ the corrective codes list $codes$ and the message $m$;

  // SECOND STEP: CORRECTION OF THE to_correct PIXELS
  $\forall i \in$ to_correct $I(i) = I_w(i) + codes(i)$;
end
```

REFERENCES

[1] Cox, I., Miller, M., Bloom, J., Fridrich, J., and Kalker, T., [*Digital Watermarking and Steganography*], ch. 11, 382, in Multimedia Information and Systems, Morgan Kaufmann, 2nd ed. (Nov. 2007).

[2] Chaumont, M. and Puech, W., "A 8-Bit-Grey-Level Image Embedding its 512 Color Palette," in [*The 16th European Signal Processing Conference, EUSIPCO'2008*], 5 pages (Aug. 2008).

[3] Bender, W., Gruhl, D., Morimoto, N., and Lu, A., "Techniques for data-hiding," in [*IBM Syst. J.*], **35**(3), 313–336 (1996).

[4] Vleeschouwer, C. D., Delaigle, J., and Macq, B., "Circular Interpretation on Histogram for Reversible Watermarking," in [*IEEE International Multimedia Signal Processecing Workshop, IMSPW'2001*], 345–350 (Oct. 2001).

[5] Fridrich, J., Goljan, M., and Du, R., "Invertible Authentication," in [*IS&T/SPIE Annual Symposium on Electronic Imaging, Security Watermarking Multimedia Contents, SPIE'2001*], **4314**, 197–208 (Jan. 2001).

[6] Celik, M. U., Sharma, G., and Tekalp, A. M., "Lossless Watermarking for Image Authentication: A New Framework and an Implementation," *IEEE Transactions on Image Processing* **15** (Apr. 2006).

[7] Tian, J., "Reversible Data Embedding Using a Difference Expansion," *IEEE Transactions on Circuits and Systems for Video Technology* **13**, 890–896 (Aug. 2003).

[8] van Leest, A., van der Veen, M., and Bruekers, A., "Reversible Watermarking for Images," in [*IS&T/SPIE Annual Symposium on Electronic Imaging, Security Watermarking Multimedia Contents, SPIE'2004*], **5306** (Jan. 2004).

[9] Ni, Z., Shi, Y.-Q., Ansari, N., and Su, W., "Reversible Data Hiding," *IEEE Transactions on Circuits and Systems for Video Technology* **16** (Mar. 2006).

[10] Coltuc, D., "Improved Capacity Reversible Watermarking," in [*IEEE International Conference on Image Processing, ICIP'2007*], (Sept. 2007).