Proc. VIIth Digital Image Computing: Techniques and Applications, Sun C., Talbot H., Ourselin S. and Adriaansen T. (Eds.), 10-12 Dec. 2003, Sydney

Reversible Data Embedding Based on the Haar Wavelet Decomposition

Henk Heijmans and Lute Kamstra

CWI, Kruislaan 413, 1098 SJ Amsterdam
The Netherlands,
`Henk.Heijmans@cwi.nl` and `Lute.Kamstra@cwi.nl`

Abstract. This paper presents an improved version of Tian's algorithm for reversible data embedding. The improvement is highest at low embedding rates.

1 Introduction

Digital information and communication technologies are now within reach of every household and they have changed our daily life in many different ways. It doesn't require too much imagination to expect that this pervasion of our society by modern technologies such as the Internet, will be a maintaining and continuing one. The impact of the Internet is perhaps most visible in the use of digital media such as audio, image, and video. It has not so much affected the creation of these media, but to a much larger extent their reproduction and distribution. Every schoolboy or schoolgirl with a computer and Internet connection is now able to set up his or her own online music store.

Digital media have numerous advantages over analog media, such as higher quality, easy editing, lossless copying, and fast and efficient distribution. At the same time, such advantages may turn into disadvantages when the underlying technologies are being exploited by non-authorised users. But digital media also offer the possibility to embed additional data into the original media data in a way that is perceptually, and sometimes also statistically, undetectable. This data embedding potential can be exploited to build protection mechanisms against the threats mentioned before, or to provide additional functionalities. In this paper we discuss one particular algorithm for data embedding which has the property that it is reversible, meaning that it is possible to restore the original content after data extraction.

In the following section we give a short overview of some data embedding applications, and in Section 3 we concentrate on the reversible (or lossless) data embedding framework. An interesting and powerful reversible data embedding algorithm was introduced by Tian [9]. We will discuss it in more detail in Section 4, and we present a major improvement of this algorithm in Section 5. Finally, Section 6 gives some experimental results and lists some conclusions.

2 Data embedding

There is a huge literature dealing with various technical, application oriented, and legal aspects of data embedding but unfortunately there is no common agreement about the use of terminology[1] in this matter. In the literature, one will also often come across terms such as data hiding, steganography, and (digital) watermarking. In this paper, we will reserve the term "watermarking" for the family of data embedding techniques that satisfy some additional constraints, in particular robustness; see also below.

We briefly list various data embedding frameworks along with some applications for which they may be used. We warn the reader that this list is far from exhaustive and is only meant to give some flavour of this research area. More comprehensive accounts can be found, among others, in [3,5,8]. In the remainder of this paper we will restrict ourselves to digital images.

Digital watermarking, as the phrase suggest, is the process of inserting an invisible watermark into digital media, e.g. for the purpose of copy control (e.g., DVD's) or copyright protection. A major constraint that any watermarking scheme needs to satisfy is *robustness*: the embedded watermark needs to be robust under unintentional distortions such as digital-to-analog and analog-to-digital conversion, enhancement or filtering, compression, etc, as well as under intentional distortions meant to remove the watermark, or rather to render it undetectable.

Fingerprinting In the literature one finds different interpretations of the word 'fingerprinting'. Firstly, it is used for the technique where unique information (think of a digital serial number) is inserted into each copy of the image. This enables the rightful owner to trace unauthorized copies back to the source. Another interpretation, that seems to become more accepted by the community is that of extracting a set of features which uniquely identifies the image. In that sense, the concept is very similar to that of human fingerprints, or some other biometric feature. The fingerprinting system consists of two components, the extraction algorithm which is used to generate a database of fingerprints and the recognition algorithm that compares extracted fingerprints against a large database of fingerprints. The technique of fingerprinting seems very useful for audio applications, where it can be used for example to recognise CD tracks, etc. Strictly speaking, fingerprinting (in the second meaning) does not belong to the family of data embedding as, in most cases, no data is inserted into the media.

Authentication of digital media is concerned with the problem of verifying that the media is the same as the one that was originally created and has not been tampered with. Additionally, one may be interested in authentication algorithms that are able to detect the nature and location of later modifications.

Steganography literally means "covered writing". It involves methods of transmitting secret messages by means of innocuous carriers in such a way that the very existence of the hidden message is undetectable; the latter means not only that is imperceptible, but also that its embedding cannot easily be revealed

[1] A first attempt in this direction has been made by Pfitzmann in [1].

by advanced statistical methods. Here, the hidden message can be any kind of data that allows a binary representation. Two major differences with watermarking applications are evident: robustness of the method under distortions is much less of an issue for steganography than for watermarking, but on the other hand, steganography requires a much higher capacity than watermarking.

Indexing is concerned with the embedding of metadata information into the media. The metadata can be provided as additional information, but can also be extracted and then inserted into the media e.g. to allow efficient browsing. A major difference with watermarking and steganography is that security of the method is not a requirement, as the embedded data provides additional functionality to the user.

As one can one see from these different type of applications, the possible list of requirements a data embedding scheme has to meet is long and diverse, and most importantly, strongly dependent on the application. Watermarking schemes for copy control or copyright protection need to be robust in the sense that they can resist attacks with the intention to remove the watermark or insert another one. For authentication applications, on the other hand, the watermark needs to be fragile. But in both cases, the watermark needs to be secure in the sense that embedding, and possibly also detection, requires a secret key. In this respect, a guiding principle, which is becoming more accepted within the watermarking community, is *Kerckhoff's principle*, known from cryptography. It says that the algorithm that is used for watermarking is publicly known and that the security of the method lies in the choice of the key. There are also cases, such as indexing, in which security may not be an issue at all. For steganography applications, the capacity of the method is a major issue, whereas in some watermarking applications it suffices to embed just one bit, indicating the presence of the watermark.[2]

3 Reversible data embedding

Most existing data-embedding algorithms distort the original signal in an irreversible manner and then one of the challenges is to minimise distortion against capacity. But there are various applications, e.g. in medical or military imaging, where any distortion, no matter how small, is intolerable. In such cases one has to take recourse to *reversible* (also called *lossless*) data embedding methods. This means that the original signal can be recovered after extraction of the message (or watermark). That such reversible embedding is possible at all, is due to the fact that images usually possess strong spatial correlations. In fact, such correlations are exploited by compression algorithms such as JPEG, to reduce their size.

In practice, the main ingredient of a reversible data embedding algorithm is the introduction of an alternative representation which, as much as possible,

[2] Note however, that the fact that only one bit is embedded does not mean that only one bit of original data has been changed by the algorithm!

decorrelates the data. In Figure 1 below, this new representation is described by a transformation R with inverse R^{-1}. When applied to an image, it creates a

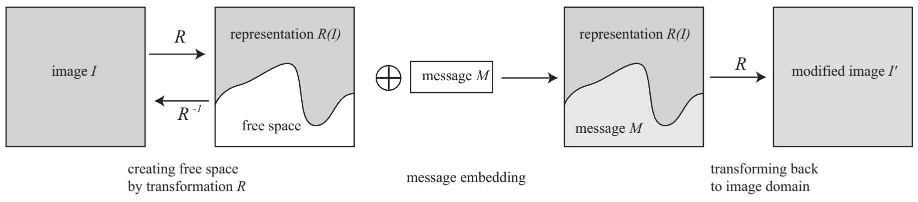

Fig. 1. *General scheme of reversible data embedding.*

certain amount of "free space" which can be used to insert the message M. After back-transformation to the original image domain we obtain a modified image I'. Symbolically:

$$I' = R^{-1}\left(R(I) \oplus M\right) .$$

Here the '$\oplus$' denotes the message embedding algorithm, which is assumed to be an invertible process with inverse denoted by '$\ominus$'. Thus we can extract both M and I at the decoder site:

$$I = R^{-1}\left(R(I') \ominus M\right) .$$

Preferably, R is chosen in such a way that it creates a large free space, i.e., has a strong decorrelating effect. Furthermore, filling the free space with random values should not lead to great distortions in the reconstructed image.

The literature on reversible data embedding is quite limited. Some interesting work is done by Fridrich and co-workers [4], by Celik *et al* [2], and by Kalker and Willems [6]. The algorithm in this paper builds upon a recent approach by Tian [9], whose key idea is to choose for R the Haar wavelet decomposition and to embed the message M in the resulting high-pass band. Tian's algorithm will be described in much greater detail in the following section

4 Tian's method

Tian [9] presented a new method for reversible data embedding which he called *difference expansion*. The key idea is to add, rather than replace, bits in the detail band of the Haar expansion. As a result, the embedding capacity for this method is considerably higher than in [2, 4].

4.1 The Haar transform

Tian's method is based on a row-by-row[3] integer-valued Haar wavelet transform of the image. Given a grey-scale image $I : [1, M_r] \times [1, M_c] \to \{0, 1, \ldots, 255\}$,

[3] This can be modified and extended in various ways.

where M_r, M_c is the numbers of rows and columns, respectively. Consider the values at two neighbouring pixels $x = I(i, 2j)$ and $y = I(i, 2j + 1)$. The integer Haar transform maps the pair (x, y) onto another pair (l, h) given by

$$l = \left\lfloor \frac{x+y}{2} \right\rfloor \quad \text{and} \quad h = x - y \,. \tag{1}$$

Here $\lfloor\cdot\rfloor$ is the floor function, i.e., $\lfloor x \rfloor$ is the largest integer $\leq x$. The transformation given by (1) is invertible and x, y can be computed from

$$x = l + \left\lfloor \frac{h+1}{2} \right\rfloor \quad \text{and} \quad y = l - \left\lfloor \frac{h}{2} \right\rfloor \,. \tag{2}$$

The Haar transform in (1) maps the original image I onto a low-pass image L given by $L(i, j) = l$ and a high-pass image H with $H(i, j) = h$. Both images have domain $D = [1, M_r] \times [1, M_c/2]$; note that we have assumed that M_c is even. Data embedding is done by modifying the values of the high-pass image H. This, however, is not always possible as it may give rise to values x, y reconstructed from (2) which lie outside the range $[0, 255]$. In fact, one can easily verify that, for a given l, the value h must lie inside a so-called *invertibility region* $R(l)$ in order that the reconstructed values x, y are integers between 0 and 255. This region is determined by

$$|h| \leq 2(255 - l) \text{ and } |h| \leq 2l + 1 \,.$$

Note that the size and shape of $R(l)$ depends heavily on l; if l is close to the endpoints 0 or 255, then it is small, but if l has some intermediate value, i.e. close to 127, then it can be large.

4.2 Partitioning of the subband domain

The domain of the subband images L and H is partitioned into four parts as shown in Figure 2. Let C be the subset of *changeable* pixels, i.e., the pixels (i, j) such that the LSB (least significant bit) of $h = H(i, j)$ can be flipped while preserving invertibility, i.e., $4\lfloor h/2 \rfloor - h + 1 \in R(l)$, where $l = L(i, j)$. Furthermore, E is the subset of *expandable* pixels (i, j): we can add a bit to h without destroying invertibility, i.e. $2h, 2h + 1 \in R(l)$. Obviously, E is a subset of C. Let E_0 be the subset of C with pixels (i, j) for which $H(i, j)$ equals -1 or 0. Then $(i, j) \in C$ iff $(i, j) \in E$, and thus we get that $E_0 \subseteq E \subseteq C$. These definitions partition the domain D into four disjoint subsets: $D = E_0 \cup (E \setminus E_0) \cup (C \setminus E) \cup (D \setminus C)$. The first two subsets comprise all expandable pixels, the third subset contains pixels that are changeable but not expandable, and the last subset contains all pixels that are not changeable. An illustration is given in Figure 2.

4.3 Embedding algorithm

Data embedding takes place by a combination of expansion and modification. Since, as a rule, expansion leads to more severe distortion than modification,

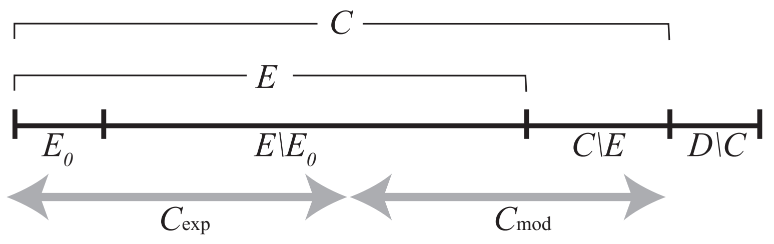

Fig. 2. *Partitioning of the domain D into four subsets.*

the number of pixels that will be subject to expansion will be limited as much as possible. We denote by C_{exp} and C_{mod} the subsets of pixels in C that will be subject to expansion and modification, respectively. Here C_{exp} is a union of E_0 and a subset of $E \setminus E_0$. There is freedom as to the choice of this subset, but it should be taken into account that small difference values give rise to small distortions in the reconstructed values x' and y'.

The subset $C_{\mathrm{mod}} = C \setminus C_{\mathrm{exp}}$ contains the remaining pixels in C. To enable reconstruction, the decoder has to know how C was subdivided into C_{exp} and C_{mod}. This is achieved by the definition of a *location map*, which, in Tian's paper [9] is a binary image on D which equals 1 at pixels in C_{exp} and 0 elsewhere. Using a compression scheme such as JBIG2, this location map is then compressed losslessly, resulting in a bitstream $\mathcal{L}$. Since for most real-world images, more than 90% of the pixels in D are expandable, one finds location maps that allow very efficient compression, resulting in a high embedding capacity.

Not only the location map has to be embedded but also the LSB's of difference values of pixels in C_{mod}, as these are overwritten at the encoding stage. We denote the corresponding bitstream by $\mathcal{C}$. Thus the overall bitstream that needs to be embedded is given by $\mathcal{B} = \mathcal{L} \cdot \mathcal{C} \cdot \mathcal{P}$, where $\mathcal{P}$ is the (message) payload, and '$\cdot$' denotes concatenation.

4.4 Discussion

The original method of Tian has two very serious drawbacks. The first concerns the capacity control problem. It is easy to see that embedding is indeed possible if $|\mathcal{L}|+|\mathcal{C}|+|\mathcal{P}| \leq |C|$,which, after using that $|\mathcal{C}| = |C_{\mathrm{mod}}|$ and $|C| = |C_{\mathrm{mod}}|+|C_{\mathrm{exp}}|$, can be rewritten as $|\mathcal{L}| + |\mathcal{P}| \leq |C_{\mathrm{exp}}|$. How does one choose C_{exp} and C_{mod} in such a way that this inequality holds and distortion is limited? A second disadvantage is the overhead cost due to the embedding of the location map bitstream $\mathcal{L}$, which always represents the entire domain D, even for very small payloads $\mathcal{P}$. As a result, small payloads are embedded at relatively high distortions. In the following section we present an alternative method which makes it easier to deal with the capacity control problem and also circumvents the second problem.

5 Modification of Tian's algorithm

Our alternative method contains the following two major improvements. We use the low-pass image L to 'predict' the location map, and use an adaptive arithmetic coder to compress the much sparser map of erroneous predictions.

5.1 Global description

As before, embedding is done by either expanding or modifying the values of the high-pass image H. Thus the encoder does not affect L and we can use the information contained in L to predict the nature of H, both at the encoder and the decoder. More precisely, we predict that $|H(i,j)|$ is small, and hence pixel (i,j) is likely to be expandable, if L is relatively homogeneous near (i,j). We introduce a *regularity measure* $\mu : C \to \mathbb{R}_+$ which measures the regularity or smoothness of L in the neighbourhood of (i,j) (regular pixels being represented by small μ-values). There are several options for defining μ, but in this paper we will restrict ourselves to a local variance measure:

$$\mu(i,j) = \frac{1}{|W(i,j)|} \sum_{(i',j') \in W(i,j)} \left(L(i',j') - \bar{L}(i,j) \right)^2 , \qquad (3)$$

where $W(i,j)$ is a window centered at (i,j), and $\bar{L}(i,j)$ is the average of L inside $W(i,j)$. We use μ to sort C into a list $C_\mu = \{(i_k, j_k)\}$ with ascending μ- values. Pixels at the beginning of the list are more likely to be expandable than pixels more toward the end, and in addition, H tends to be small at such pixels. The order of embedding follows that of the list C_μ and the location map merely represents the correctness of our prediction. To be precise, the location map, or rather, *location bitstream*, is the sequence $a_1 a_2 \cdots$, where $a_k = 0$ if $(i_k, j_k) \in E$ and $a_k = 1$ if $(i_k, j_k) \in C \backslash E$. It can be expected that the location bitstream consists mostly of 0's, especially in the beginning, and therefore allows strong lossless compression, e.g., by an adaptive arithmetic coder (AAC).

5.2 Capacity control

Denote by cap(n) the net capacity in case that the first n pixels in C_μ are used for embedding. For embedding the payload $\mathcal{P}$, we need to choose n large enough, namely so that cap$(n) \geq |\mathcal{P}|$. Denote by $\mathcal{L}(n)$ the encoded (i.e., compressed) location bitstream, and by $\mathcal{C}(n)$ the bits that are overwritten. Thus

$$\mathcal{B}(n) = \mathcal{L}(n) \cdot \mathcal{C}(n) \cdot \mathcal{P} ,$$

is the bitstream that is eventually embedded. Denote by $\kappa(n) = |\mathcal{L}(n)|$ the length of the bitstream $\mathcal{L}(n)$ that results from the AAC encoding of the location bitstream $a_1 a_2 \cdots a_n$. Let

$$\sigma_1(n) = \sum_{k=1}^{\kappa(n)} a_k = |\mathcal{C}(n)|$$

be the number of 1's in the first $\kappa(n)$ bits of the location bitstream. Such bits correspond with pixels which are changeable but not expandable, hence they are overwritten and stored in $\mathcal{C}(n)$; therefore $\sigma_1(n) = |\mathcal{C}(n)|$. The number of 1's in the remaining part of the location map equals

$$\sigma_2(n) = \sum_{k=\kappa(n)+1}^{n} a_k \,.$$

To keep distortion minimal, we would prefer to skip pixels that are not expandable rather than overwriting them. However, skipping a pixel is only an option if the decoder is aware of its non-expandability; this is only the case *after* the location bitstream has been decoded, i.e., if $k \geq \kappa(n)$. Therefore $\sigma(n) = \sigma_1(n)+\sigma_2(n)$ pixels among the first n are non-expandable, and hence ineffective. Thus we arrive at the identity $\text{cap}(n) = n - \kappa(n) - \sigma(n)$, and hence $\text{cap}(n) \geq |\mathcal{P}|$ is satisfied if $|\mathcal{P}| \leq n - \kappa(n) - \sigma(n)$. In general, $\text{cap}(n)$ is increasing with n. For small n it will be negative due to some overhead such as the header comprising κ_0 bits.

5.3 Encoding algorithm

In this and the following subsection we describe the actual encoding and decoding algorithm. Here we do not have enough room to give all the details. The reader interested in these details is referred to [7].

The input for the encoding algorithm is an input image I along with a payload $\mathcal{P}$ that needs to be embedded into I. The output is a modified image I' with the same dimensions as I. The algorithm starts with the Haar decomposition of image I into L and H. Now we can also compute the partition of the domain shown in Figure 2. Next, we compute the regularity measure μ from L, the sorted list C_μ from C and μ, and the location bitstream $a_1 a_2 \cdots$. This bitstream is compressed with an adaptive arithmetic coder (AAC) till the requested capacity is available, i.e., $\text{cap}(n) \geq |\mathcal{P}|$. This yields the output bitstream $b_1 b_2 \cdots b_{\kappa(n)}$[4] which is then embedded in the first $\kappa(n)$ pixels of C_μ, either by expansion (if $a_k = 0$) or by modification (if $a_k = 1$). In the latter case, the LSB of $H(i_k, j_k)$ has to be stored in order to enable inversion at the decoder. We obtain the correction bitstream $\mathcal{C}(n)$ to which we concatenate the payload $\mathcal{P}$, and embed this at the remaining pixels of C_μ, or rather $C_\mu \cap E$. In other words, pixels for which $a_k = 1$ are skipped. Finally, we end up with a modified high-pass image H', and now I' is found by taking the inverse Haar transform of L and H'.

5.4 Decoding algorithm

The decoding algorithm extracts both the original input image I and the payload $\mathcal{P}$ from the modified image I'. First, we apply the Haar transform to I' to obtain L and H'; note that the low-pass band L is the same as for I, and that the high-pass band H' contains all embedded data. We can use L to compute the

[4] This includes a fixed header used to store the length of the body part; see [7].

regularity measure μ that was used for embedding. Since the changeable pixels haven't been affected by the embedding either, we are able to recover the sorted list C_μ. Thus we can read the bitstream $b_1 b_2 \cdots b_{\kappa(n)}$ which, after AAC-decoding, returns the location bitstream $a_1 a_2 \cdots a_n$. Now we can read the remaining bits $\mathcal{C}(n) \cdot \mathcal{P}$, and restore the original H-values by means of $h = \lfloor h'/2 \rfloor$. Since at this stage, we have the location map at our disposal, we can simply skip pixels (i_k, j_k) in $C \backslash E$, characterized by $a_k = 1$. In a next step we restore the H-values at pixels with $k = 1, 2, \ldots, \kappa(n)$: if $a_k = 0$ then $h = \lfloor h'/2 \rfloor$, otherwise $h = 2 \lfloor h'/2 \rfloor + b$, where b was the bit overwritten by the encoder and stored in the correction bitstream $\mathcal{C}(n)$. Now the remainder of the bitstream is $\mathcal{P}$. Applying the inverse Haar transform to L and H returns the original image I.

6 Experimental results and discussion

We tested the original method of Tian and our variant of his method on a number of test images and compared the distortion (PSNR) versus capacity behaviour of both methods. Here, we will present the results of this comparison for one image only: the 512×512 Lenna image. However, the shape of the distortion-capacity curves of other images is very similar. The maximum capacity is usually the same (0.5 bits per pixel), but the distortion at which this maximum is obtained varies from image to image. Tian presented two methods for selecting

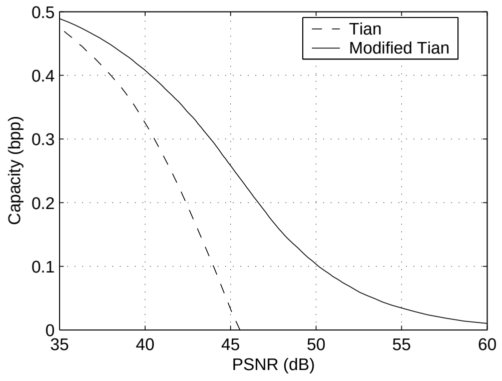

Fig. 3. *The capacity versus the distortion (right) of both Tian's original method and our variant for the Lenna image (left).*

C_{exp}, the pixels used for difference expansion. In our experiments, we used the one that minimises the mean square error introduced by the embedding. This method gives the best results when PSNR is used to measure distortion. In his experiments, Tian used a JBIG2 encoder to compress the location map. Since we did not have such an encoder at our disposal, we used an adaptive arithmetic coder in our implementation of Tian's original method. This compression method gives slightly worse results, but the difference is not substantial.

When testing our variant of Tian's algorithm, we used a number of different regularity measures μ to choose which pixels to use for expansion, most of which performed equally well. The results below are based on the local variance measure as defined in (3) with window:

$$W(i,j) = \{(i-1,j),(i,j-1),(i,j),(i,j+1),(i+1,j)\}.$$

The partition of the subband domain of Lenna is as follows: 21893 pixels ($\approx$16.70%) in E_0, 109173 pixels ($\approx$83.29%) in $E \setminus E_0$, 6 pixels in $C \setminus E$ ($\approx$0.002%), and no pixels in $D \setminus C$. Figure 3 shows, for both methods, the distortion caused by the embedding of different sized payloads.

For the Tian method, due the overhead of embedding the entire location map, the net capacity only becomes positive after selecting more than 70% of the locations for expansion, and at this point, the PSNR has already been reduced to 45.5 dB. Our modification of Tian's method does not have this problem: the net capacity is positive when less then 1% of the pixels is selected for expansion. This is clearly visible in the figure: at a capacity of 0.04 bit per pixel, the difference in distortion is 10 dB. Overall, our extension of Tian's method performs better that the original method for all capacities (even when JBIG2 compression is used in Tian's method). However, differences become smaller for higher payloads: when more pixels are selected for expansion in Tian's method, the location map can be compressed better, thus decreasing the overhead.

References

1. Anderson, R. J., Ed. *Information hiding terminology* (1996), vol. 1174 of *Lecture Notes in Computer Science*, Springer.
2. Celik, M. U., Sharma, G., Tekalp, A. M., and Saber, E. Lossless generalized-lsb data embedding. submitted to IEEE Trans. Image Proc., 2002.
3. *et al*, W. B. Applications for data hiding. *IBM Systems Journal 39*, 3&4 (2000), 547–568.
4. Fridrich, J., Goljan, M., and Du, R. Lossless data embedding - new paradigm in digital watermarking. *EURASIP J. Appl. Signal Processing: Special Issue on Emerging Applications of Multimedia Data Hiding*, 2 (2002), 185–196.
5. I. J., C., Miller, M., and Bloom, J. *Digital Watermarking.* Morgan Kaufmann Publishers, San Francisco, 2001.
6. Kalker, A. A. C. M., and Willems, F. M. J. Capacity bounds and code constructions for reversible data-hiding. In *IS&T/SPIE's 15th Ann. Symp. Electronic Imaging* (Santa Clara, California, 2003).
7. Kamstra, L., and Heijmans, H. Lossless data embedding using wavelets, 2003.
8. Swanson, M. D., Kobayashi, M., and Tewfik, A. H. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE 86*, 6 (1998), 1064–1087.
9. Tian, J. Reversible data embedding and content authentication using difference expansion. submitted to IEEE Transaction on Circuits and Systems for Video Technology, 2003.