Applying Informed Coding and Embedding to Design a Robust, High Capacity Watermark.

Matt L. Miller, Gwenaël J. Doërr, Ingemar J.Cox

Abstract

We describe a new watermarking system based on the principles of informed coding and informed embedding. This system is capable of embedding 1380 bits of information in images with dimensions 240×368 pixels. Experiments on 2000 images indicate the watermarks are robust to significant valumetric distortions, including additive noise, low pass filtering, changes in contrast, and lossy compression.

Our system encodes watermark messages with a modified trellis code in which a given message may be represented by a variety of different signals, with the embedded signal selected according to the cover image. The signal is embedded by an iterative method that seeks to ensure the message will not be confused with other messages, even after addition of noise. Fidelity is improved by the incorporation of perceptual shaping into the embedding process. We show that each of these three components improves performance substantially.

Keywords

Watermarking, informed coding, informed embedding, perceptual shaping, high capacity, robustness, dirty-paper code.

I. Introduction

In recent years, several researchers [8], [2], [4] have recognized that watermarking with blind detection can be modeled as communication with side-information at the transmitter [19]. This realization has led to the design of algorithms for *informed coding* and *informed embedding*. In informed coding, a watermark is represented with a codeword that is dependent on the cover Work. In informed embedding, each watermark pattern is tailored according to the cover Work, attempting to attain an optimal trade-off between estimates of perceptual fidelity and robustness. The reader is directed to [7] for a detailed discussion of these concepts.

To date, our own studies of informed coding and embedding have been limited to watermarks with very small data payloads [13]. Other researchers [2], [4], [18], [9] have employed informed coding to embed large data payloads in images – on the order of 1000 or more bits – but their methods involve only simple forms of informed embedding. Furthermore, as most of these methods are based on some form of lattice quantization, they exhibit only limited robustness against simple valumetric scaling, such as changes in image contrast or audio volume.

In this paper, we develop an algorithm for embedding of robust image watermarks with large data payloads using informed embedding and coding techniques.

We begin, in Section II by describing an informed embedding algorithm. Experimental results indicate that this algorithm can embed trellis-coded, 1380-bit watermarks into 240×368 images with acceptable effectiveness. By contrast, in the same context, a simple blind embedder, using the same code and fidelity, almost always fails to embed these watermarks. of blind, additive embedding (that is, it succeeds in embedding the watermark ten times as often).

Section III then proceeds to develop a method of informed coding based on a simple modification of a trellis code. When combined with our informed embedding algorithm, this results in a further substantial increase in effectiveness. Unfortunately, images watermarked with these methods exhibit an unacceptable loss of fidelity.

The fidelity problem is reduced in Section IV, where we incorporate perceptual shaping, based on Watson's perceptual distance measure [23]. This reduces the perceptual distance between watermarked and unmarked images by a factor of three, as measured by Watson's model, without substantially changing effectiveness or robustness.

Section V presents several experiments showing that, even after rather severe addition of noise, filtering, valumetric scaling, or lossy compression, all 1380 bits are correctly detected in at least 80% of watermarked images.

Matt L. Miller and Ingemar J. Cox are with the NEC Research Institute, Princeton, NJ, 08540 USA.
Gwenaël J. Doërr is with the Eurécom Institute, Sophia-Antipolis, France.

II. Informed embedding

Figure 1 illustrates the basic idea of informed embedding. Here, we view watermark embedding as a three-step

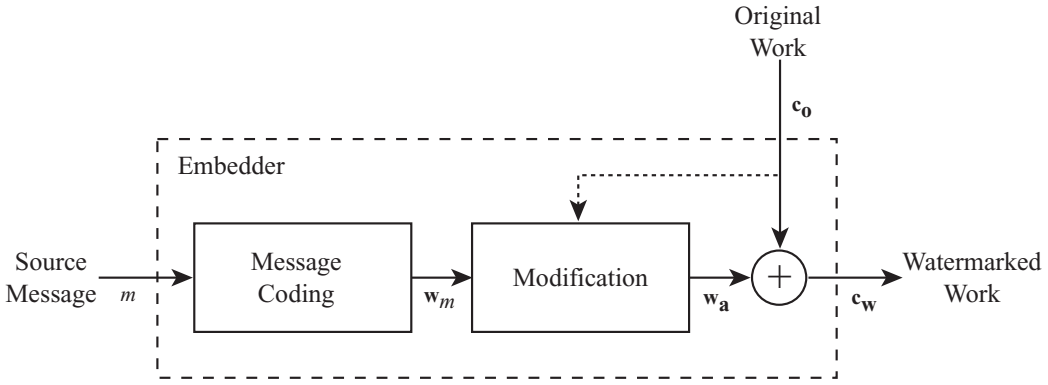

Fig. 1. Watermarking with informed embedding.

process. First, the message to be embedded is encoded as a signal, $\mathbf{w}_m$. Second, the signal is modified in preparation for embedding, yielding a modified signal, $\mathbf{w_a}$. Finally, the modified signal is added to the cover image, $\mathbf{c_o}$, to obtain the watermarked image, $\mathbf{c_w}$. In *blind embedding*, the modification step is performed independently of the cover image; it is usually just a simple, global scaling. In *informed embedding*, by contrast, the modification is a function of the image and the message signal.

Since complete information about the cover image is available, an informed embedder has complete control over the final, watermarked image. That is, it can select *any* image as $\mathbf{c_w}$ by letting $\mathbf{w_a} = \mathbf{c_w} - \mathbf{c_o}$. The task is to find an image that satisfies two conflicting criteria: 1) $\mathbf{c_w}$ should be similar enough to $\mathbf{c_o}$ to be perceptually indistinguishable, and 2) $\mathbf{c_w}$ should be close enough to $\mathbf{w}_m$ to be detected as containing the watermark, even after distortion by subsequent processing.

In practice, an informed embedding algorithm can be implemented using methods for estimating perceptual distance and watermark robustness. The algorithm may then attempt to either

1. maximize the estimated robustness while keeping a constant perceptual distance, or
2. minimize the perceptual distance while keeping a constant robustness.

We believe that most watermarking applications are best served by embedders that maintain constant robustness, and our proposed algorithm is therefore designed using the second constraint.

In order to describe the watermark embedding algorithm, we first describe the detection algorithm in Section II-A. In Section II-B, we then define a measure of robustness. Sections II-C and II-D describe two embedding methods that seek to obtain a specific value of this robustness measure while keeping the mean squared error low between the original and watermarked Works. The first of these two methods is general, and can be applied with a variety of different detection algorithms. The second is specific to the detector described in Section II-A, and is substantially faster than the general method. Section II-E describes an experiment showing that this second embedding method yields substantially better results than a simple, blind embedder.

A. Detection algorithm

Our watermarking system is built around a trellis-code, as illustrated in Figure 2. This code is similar to that used in the E_TRELLIS8/D_TRELLIS8 watermarking system of [7]. Each path through this trellis, originating at node A0,

represents a specific message. Since two arcs exit each node, there are 2^L possible paths, where L is the length of the paths. Thus, the system encodes L bits. In our experiments, $L = 1380$.

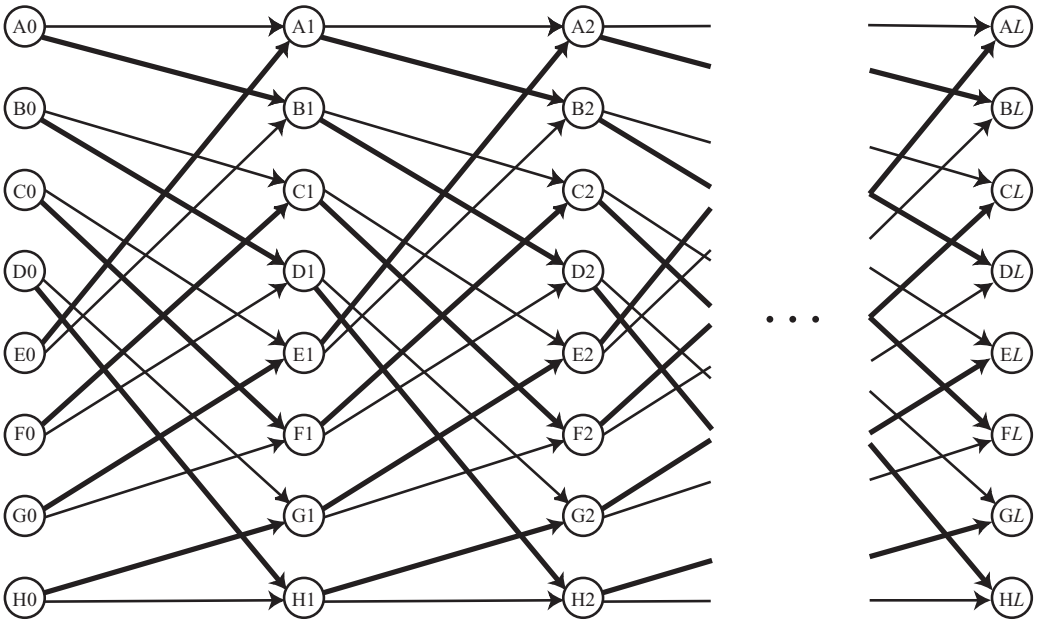

Fig. 2. Simple, 8-state trellis. Each possible message corresponds to a path from node A0 (state A at time 0) to one of the nodes at the right (any state at time L). We refer to the transition from one column of nodes to the next column of nodes as a *step*, and each such step corresponds to one bit in the coded message. A bold arc is traversed if the corresponding bit is a 1, a non-bold arc is traversed if the corresponding bit is a 0.

Each arc in the trellis is labelled with a randomly-generated, length N reference vector. In principle, these labels can be arbitrary, but for efficiency we use the same set of labels in each step of the trellis. Thus, for example, the label for the arc from node A0 to A1 has the same label as the arc from node A1 to A2, and so on. Each path, and thus each message, is coded with a length $L \times N$ vector that is the concatenation of the labels for the arcs it contains. In our experiments, $N = 12$.

Note that the paths for two different messages always differ in at least four arcs. To see this, consider one message that is encoded with the path connecting all the A nodes (A0, A1, etc.), and another message in which the first arc is from A0 to B1. The shortest way that this second message path can get back to state A takes 3 additional steps: B1 to C2, C2 to E3, and E3 to A4. Thereafter, the rest of the path can be identical to that for the first message. Thus, these two messages differ in at least four arcs. A little reflection will show that all pairs of messages are similarly distinct. This means that, even if two messages differ in only a single bit, their representations will differ substantially, and they will be unlikely to be confused as a result of added noise.

The detection algorithm proceeds as follows:

1. Convert the image into the 8×8 block-DCT domain.
2. Place all the low-frequency AC terms of the blocks into a single, length $L \times N$ vector, in random order. We refer to this as the *extracted vector*. The DCT terms used are shown in Figure 3.
3. Use a Viterbi decoder [21] to identify the path through the trellis whose $L \times N$ vector has the highest correlation with the extracted vector.
4. Identify the message that is represented by the highest-correlation path.

Note that this detection algorithm does not attempt to determine whether or not the image contains a watermark.

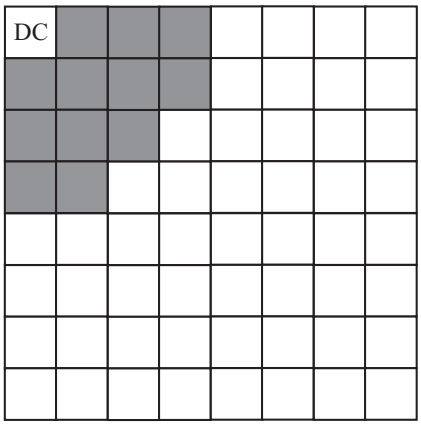

Fig. 3. DCT coefficients used by watermarking system. The upper-left corner of this figure corresponds to the DC term of an 8×8 block's DCT. The 12 coefficients shaded in gray indicate the low-frequency AC terms used.

It simply maps every possible image into an L-bit message, regardless of whether the image has had a watermark embedded. In large payload applications it is usually not important for the detector to determine whether a watermark is present, since most combinations of bit values are not meaningful. For example, suppose we use our system to embed strings of 172 ascii characters. An unwatermarked image will yield an unintelligible string, so we can easily recognize it as unwatermarked when the string is displayed.

Alternatively, if we need the detector to determine presence of watermarks, we can use some number of bits to contain an error detection checksum or signature of the message. If the signature does not match the message, the detector announces that there is no watermark. This reduces the payload by a small amount, but it yields a detector with an easily predicted false positive probability – if we use, say, 20 bits for the signature, then the probability of a false positive is 2^{-20} [7].

B. Estimate of robustness

In [13], [8], we defined an estimate of robustness suitable for small-payload watermarking systems that use correlation coefficient to test whether or not a mark is present. This was based on the amount of additive white Gaussian noise that could be added to a watermarked image before it is expected to fall outside the detection region. Since the present detector does not use correlation coefficient, or test for the presence of a watermark, this measure of robustness is not appropriate. Instead of estimating the likelihood that a watermarked image will be detected as *unwatermarked*, we need to estimate the likelihood that it will be detected as containing the *wrong message*.

To develop our new robustness measure, consider a simple system in which there are only two possible messages, represented by two different vectors. We shall denote one of the vectors $\mathbf{g}$, and the other $\mathbf{b}$. When presented with an image, $\mathbf{c}$, the detector returns the message associated with $\mathbf{g}$ if $\mathbf{g} \cdot \mathbf{c} > \mathbf{b} \cdot \mathbf{c}$, where $\mathbf{g} \cdot \mathbf{c} = \sum_i \mathbf{g}[i]\mathbf{c}[i]$ is the correlation between $\mathbf{g}$ and $\mathbf{c}$.

Suppose we wish to embed $\mathbf{g}$ into an image $\mathbf{c_o}$ (so $\mathbf{g}$ is the "good" vector – the one we want to embed – and $\mathbf{b}$ is a "bad" vector – one we do not want the watermarked image to be confused with). Our task is to estimate the chances that a proposed watermarked image, $\mathbf{c_w}$, will, after corruption by subsequent processing, be detected as containing the message $\mathbf{g}$ rather than the message $\mathbf{b}$. More precisely, we need a value that is monotonically related to the probability that message $\mathbf{g}$ will be correctly detected in a corrupted version of the watermarked Work, $\mathbf{c_w}$.

As in [13], [8], we proceed by assuming that the distortions applied to the Work after watermark embedding can be

modeled as the addition of white Gaussian noise[1]. Thus, we assume that the detector will receive $\mathbf{c_{wn}} = \mathbf{c_w} + \mathbf{n}$, where $\mathbf{n}$ is a length $L \times N$ vector whose elements are drawn independently from a Gaussian distribution with variance $\sigma_{\mathbf{n}}^2$. The probability that $\mathbf{g}$ will be detected in $\mathbf{c_{wn}}$ is

$$\begin{aligned} P\{\mathbf{g}\cdot\mathbf{c_{wn}} > \mathbf{b}\cdot\mathbf{c_{wn}}\} &= \\ P\{\mathbf{g}\cdot(\mathbf{c_w}+\mathbf{n}) > \mathbf{b}\cdot(\mathbf{c_w}+\mathbf{n})\} &= \\ P\{(\mathbf{g}-\mathbf{b})\cdot\mathbf{c_w} > (\mathbf{b}-\mathbf{g})\cdot\mathbf{n}\} &= \\ P\left\{\frac{(\mathbf{g}-\mathbf{b})\cdot\mathbf{c_w}}{|\mathbf{g}-\mathbf{b}|} > \sigma_{\mathbf{n}} r\right\} & \end{aligned} \tag{1}$$

where r is a random scalar value drawn from a unit-variance, Gaussian distribution. Clearly, the larger the value of

$$R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) = \frac{(\mathbf{g}-\mathbf{b})\cdot\mathbf{c_w}}{|\mathbf{g}-\mathbf{b}|} \tag{2}$$

the higher the probability that it will be greater than $\sigma_{\mathbf{n}} r$. Thus, the larger $R_0()$ is, the greater the chances that the watermark $\mathbf{g}$ will be correctly detected in $\mathbf{c_{wn}}$. $R_0()$, then, can serve as our robustness measure for a simple, two-message watermarking system.

To extend this measure to larger payloads, we take the minimum of R_0 over *all* possible erroneous message vectors, $\mathbf{b_1}...\mathbf{b}_{2^L-1}$. Thus,

$$R(\mathbf{c_w}, \mathbf{g}) = \min_{i=1}^{2^L-1} R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}_i). \tag{3}$$

Figure 4 illustrates a geometric interpretation of the embedding region that results when we specify that $R(\mathbf{c_w}, \mathbf{g})$ must be greater than or equal to a given value. The figure shows a Voronoi diagram representing the detection regions for various messages. By specifying a minimum value for $R(\mathbf{c_w}, \mathbf{g})$, we are insisting that $\mathbf{c_w}$ must lie a certain distance from the edge of the detection region for $\mathbf{g}$. The figure also illustrates the behavior of an ideal embedder using this robustness measure. The open circle indicates an unwatermarked cover image, and the solid dot shows the closest possible watermarked image with acceptable robustness.

C. General embedding algorithm

In practice, it is difficult to implement an algorithm to find the optimal watermarked image, as illustrated in Figure 4. Instead, we use a suboptimal, iterative algorithm. Here we present a general version of this algorithm that can be used with a wide variety of watermark coding schemes. We first present a basic outline of the algorithm that is independent of the specific definition of R_0. We then present a practical, Monte Carlo version of the algorithm, that relies on R_0 being a measure of robustness against white noise. A version specifically designed for the trellis-coded watermarks we have implemented is presented in the next section.

We assume that we have a black-box watermark encoder, $W(m)$, that maps a sequence of bits, m, into a watermark signal, $\mathbf{w}_m$. We further have a black-box watermark detector, $D(\mathbf{c})$, that maps an image, $\mathbf{c}$, into the sequence of bits corresponding to the watermark signal with which the image has the highest correlation. We make no assumptions about how these two functions work internally.

We now give the basic outline of the iterative embedding algorithm. Given a cover image, $\mathbf{c_o}$, a message to embed m, and a target robustness value R_t (i.e. a target value for $R()$ from Equation 3), the algorithm proceeds as follows:

1. Let $\mathbf{g} = W(m)$ and $\mathbf{c_w} = \mathbf{c_o}$.
2. Find the signal $\mathbf{b} \neq \mathbf{g}$ that minimizes $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b})$.

[1]See [7] for a justification of this assumption.

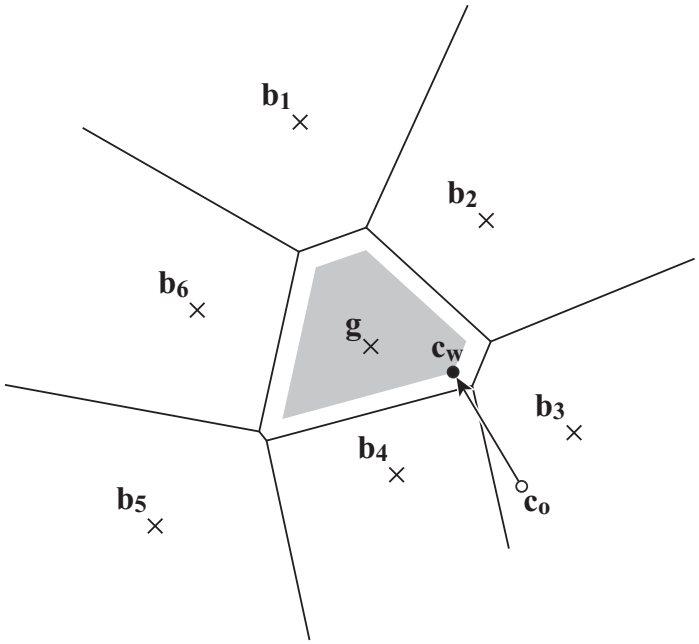

Fig. 4. Geometric interpretation of embedding region defined by placing a lower limit on $R(\mathbf{c_w}, \mathbf{g})$. Each point in this space corresponds to a possible image. The Voronoi diagram indicates the detection regions for seven different message vectors, which are indicated with X's. Six of these are "bad" vectors we do not want to embed. The seventh, $\mathbf{g}$, is the "good" vector we intend to embed. The gray region indicates the set of images which satisfy the constraint on $R()$. The open circle, the arrow, and the solid dot illustrate the behavior of an ideal embedder – cover image $\mathbf{c_o}$ would be moved to the closest point in the embedding region, $\mathbf{c_w}$.

3. If $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) \geq R_t$, then terminate.
4. Otherwise, modify $\mathbf{c_w}$ so that $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) = R_t$, and go to step 2.

The modification of $\mathbf{c_w}$ in step 4 is performed as follows:

$$\begin{aligned} \mathbf{d} &= \frac{\mathbf{g} - \mathbf{b}}{|\mathbf{g} - \mathbf{b}|} \\ \alpha &= R_t - R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) \\ \mathbf{c_w} &\leftarrow \mathbf{c_w} + \alpha \mathbf{d} \end{aligned} \tag{4}$$

The new $\mathbf{c_w}$ yields $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b})$ exactly equal to R_t, while having a minimum Euclidian distance from the previous $\mathbf{c_w}$.

The operation of this algorithm is shown geometrically in Figure 5. In the first iteration, $\mathbf{c_w}$ lies in the detection region for $\mathbf{b_3}$, so $\mathbf{b} = \mathbf{b_3}$ in step 2, and $\mathbf{c_w}$ is moved to a point beyond the boundary between $\mathbf{g}$ and $\mathbf{b_3}$. In the second iteration, $\mathbf{b} = \mathbf{b_4}$, and $\mathbf{g}$ is moved into the detection region for $\mathbf{g}$. In the final iteration, the closest bad vector is still $\mathbf{b_4}$, but $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b_4})$ is already satisfactory, so the algorithm terminates. The figure clearly illustrates that this algorithm is suboptimal, since it does not yield the optimal point identified in Figure 4. Nevertheless, it is practical to implement.

The identification of $\mathbf{b}$ in step 2 depends on the method of coding. For most codes, it is not easy. We therefore apply a simple, Monte Carlo approach by letting $\mathbf{b} = W(D(\mathbf{c_w} + \mathbf{n}))$, where $\mathbf{n}$ is some random noise. If a small amount of noise is added to $\mathbf{c_w}$, and the detector returns a message other than m, then $\mathbf{b}$ is likely to yield a low value of $R_0()$. If there exist any vectors that yield values of $R_0()$ below the target value, R_t, then $\mathbf{b}$ is likely to be one of them.

The best amount of noise to add changes as the embedding process progresses. In the first iteration, when $\mathbf{c_w} = \mathbf{c_o}$, it is unlikely that $D(\mathbf{c_w}) = m$, so we need not add any noise at all to find the nearest bad vector. This will remain true through several iterations, until $D(\mathbf{c_w}) = m$. At that point, the closest bad vectors are likely to yield very low values of $R_0()$, so we need add only a small amount of noise to find them. As $\mathbf{c_w}$ is modified to be robust against confusion with

these vectors, the remaining vectors yield higher values of $R_0()$, and thus require the addition of more noise. In general, if too little noise is added, $W(D(\mathbf{c_w} + \mathbf{n}))$ will equal $\mathbf{g}$. If too much noise is added, $W(D(\mathbf{c_w} + \mathbf{n}))$ has a high chance of producing a vector for which $R_0()$ is much larger than the minimum available value.

We therefore dynamically adjust the amount of noise added in each iteration. At the beginning, the standard deviation of the noise, $\sigma_\mathbf{n}$, is 0, so no noise is added. Whenever $W(D(\mathbf{c_w} + \mathbf{n}))$ yields $\mathbf{g}$, we increase $\sigma_\mathbf{n}$ by a small, fixed amount, δ. When $W(D(\mathbf{c_w} + \mathbf{n}))$ yields a bad vector, $\mathbf{b}$, but $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b})$ is greater than or equal to R_t, we decrease $\sigma_\mathbf{n}$ by δ. If $W(D(\mathbf{c_w} + \mathbf{n}))$ yields a bad vector, $\mathbf{b}$, and $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) < R_t$, we modify $\mathbf{c_w}$ and leave δ unchanged. In our experiments, $\delta = 0.1$.

Since this Monte Carlo approach does not guarantee that we find the $\mathbf{b}$ that minimizes $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b})$ in each iteration, we cannot terminate the algorithm the first time that $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b})$ is greater than or equal to the target value – there might still be some other $\mathbf{b}$ for which $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) < R_t$. We therefore maintain a count of the number of consecutive $\mathbf{b}$'s found for which $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) \geq R_t$. The algorithm terminates when this count reaches a specified limit. In our experiments, the limit was set at 100.

Thus, the complete, general version of our informed embedding algorithm proceeds as follows:

1. Let $\mathbf{g} = W(m)$, $\mathbf{c_w} = \mathbf{c_o}$, $\sigma_\mathbf{n} = 0$, and $j = 0$.
2. Let $\mathbf{b} = W(D(\mathbf{c_w} + \mathbf{n}))$, where $\mathbf{n}$ is a random vector with each element drawn independently from a Gaussian distribution with variance $\sigma_\mathbf{n}^2$.
3. If $\mathbf{b} = \mathbf{g}$, let $\sigma_\mathbf{n} \leftarrow \sigma_\mathbf{n} + \delta$ and go back to step 2.
4. If $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) < R_t$, then modify $\mathbf{c_w}$ according to Equation 4, reset j to 0, and go back to step 2.
5. If $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) \geq R_t$, then increment j. If $j < 100$, then let $\sigma_\mathbf{n} \leftarrow \sigma_\mathbf{n} - \delta$ and go back to step 2. Otherwise, terminate.

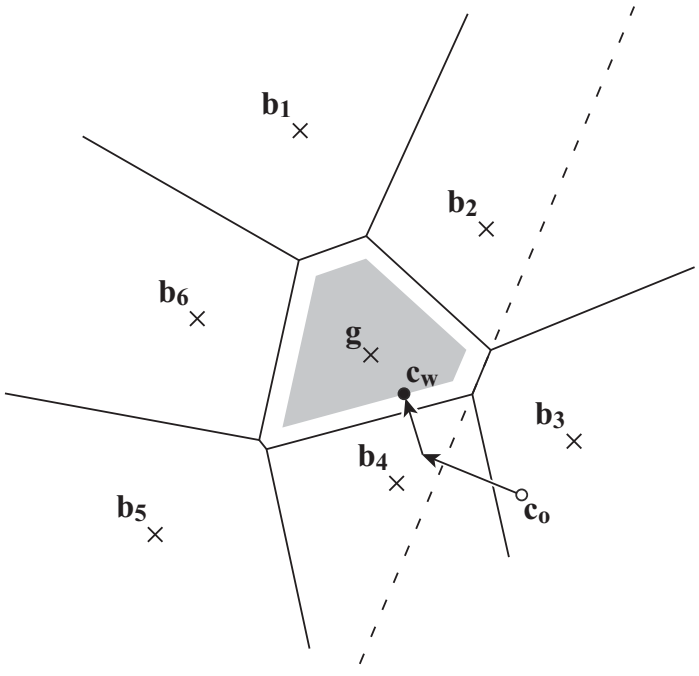

Fig. 5. Geometric interpretation of a sub-optimal, iterative method for informed embedding. In each iteration, the cover image is modified to prevent it from being decoded as one "bad" vector. In this illustration, the first iteration modifies $\mathbf{c_o}$ so that it will not be detected as containing message $\mathbf{b_3}$ (even after addition of some noise). The second iteration modifies it so it will not be detected as containing $\mathbf{b_4}$. This results in a watermarked image, $\mathbf{c_w}$, within the embedding region around $\mathbf{g}$.

D. Specific embedding algorithm for trellis-codes

The general method outlined above is very slow, as it can often take many thousands of iterations to terminate. When implemented with our trellis-coded watermarking system, each iteration requires running the Viterbi decoder on the entire extracted vector (sequence of low-frequency DCT coefficients). This, in turn, requires performing $L \times A$ length N correlations, where A is the number of arcs in each step of the trellis.

Thus, instead of adding noise to $\mathbf{c_w}$ and running the detector in each iteration, we use a modified version of the Viterbi decoder that produces probabalistic results. Normally, the Viterbi algorithm maintains a table that indicates the correlation between the extracted vector, $\mathbf{v}$, and the vectors for the paths up to all the states in a given step of the trellis (see [7] for a description of the Viterbi algorithm in these terms). Our modified decoder adds a random number to each value in this table before proceeding to the next step of the trellis. This means that the decoder might return a path other than the one that yields the highest correlation with $\mathbf{v}$.

The behavior of the modified Viterbi decoder is similar to the results of adding noise to $\mathbf{c_w}$ before running the detector, but it is not identical. Nevertheless, in informal tests, we found the performance of our informed embedder to be unaffected by the difference. By using the modified Viterbi decoder, we can significantly reduce running time because the correlations for the arcs of the trellis need not be recomputed every time the detector is used. Instead, they need only be recomputed when $\mathbf{c_w}$ is modified. As each of these correlations compares two length-N vectors, eliminating these operations reduces the running time by almost a factor of N.

E. Performance

To examine the effectiveness of the informed embedding algorithm described here, we used it to embed 1380-bit watermarks into two thousand, 240×368-pixel images. We then applied the detector to them, with no intervening processing, and measured the number of message errors (i.e. the number of times that at least one of the 1380 bits was incorrectly detected).

In theory, the algorithm should be 100% effective (0% message error rate), since it does not terminate before $D(\mathbf{c_w}) = m$. However, our implementation operates on the extracted vector, rather than on the image itself. That is, we begin by extracting a vector from the block DCT of $\mathbf{c_o}$ in the same manner as the detector. We then use the embedding algorithm to modify this vector so that it has the desired message embedded. Finally, we put the elements of the modified vector back into their respective DCT coefficients, and convert the image back to the spatial domain. This is mathematically equivalent to applying the embedding algorithm to the entire image, but round-off and clipping introduce the possibility of subsequent decoding errors.

For purposes of comparison, we also implemented a simple, blind embedding algorithm. This used the same watermark extraction process (same DCT coefficients and ordering), data payload, and trellis code, as the informed embedder. However, in the blind embedder, the watermarked image was computed as $\mathbf{c_w} = \mathbf{c_o} + \alpha W(\mathbf{m})$, where α is a constant controlling the *embedding strength*. We chose α to produce roughly the same fidelity impact as that produced by the informed embedding algorithm. The degradation in fidelity was measured using Watson's model ([23]). The results of both experiments are shown in Table I.

Clearly, for the same average fidelity impact, our informed embedder is far more effective than blind embedding. However, it must be noted that the perceptual impact of the algorithm is unacceptable, as measured by Watson's model and illustrated in Figure 7. This problem is dealt with in subsequent sections.

Fig. 6. Original cover image. (Note: we chose this image for our figures because it exhibited typical behaviour in our various experiments, and it shows the perceptual differences between the embedding algorithms we present.)

TABLE I
BLIND EMBEDDING VERSUS INFORMED EMBEDDING.

	Watson distance	**MER**
Blind embedding	200.04	99.85%
Informed embedding	201.06	12.5%

III. INFORMED CODING

The embedding algorithm of the previous section uses information contained in the cover image during the modification stage. However, each message is represented by a unique codeword that is independent of the image. Research in communications with side-information at the embedder suggests that better results can be obtained if the coding process itself is a function of the cover image. Therefore, we now consider *informed coding*, in which each message is mapped into a set of alternative codewords and the choice of which codeword to embed is determined by information contained in the cover image. This is illustrated in Figure 8.

A. Dirty-paper codes

The use of informed coding in watermarking was inspired by the results of Costa [6]. Costa studied the capacity of a communications channel that consists of two additive, white Gaussian noise sources, the first of which is perfectly known to the transmitter, while the receiver has no knowledge of either (see Figure 9). He described this channel using the analogy of writing on dirty paper. Imagine a sheet of paper covered with a Normally-distributed pattern of "dirt". This dirt is the first noise source, which the transmitter can examine. The transmitter writes a message on this paper and sends it to a receiver. Along the way, the paper may acquire a second layer of "dirt", corresponding to the second

Fig. 7. Result of iterative informed embedding algorithm. Watermark noise is clearly visible, particularly in the sky.

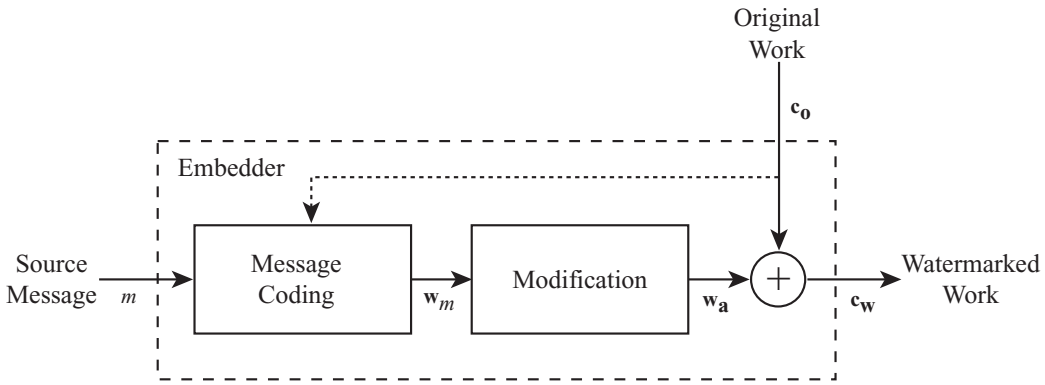

Fig. 8. Watermarking with informed coding.

noise source. The receiver cannot distinguish between dirt and the ink used to write the message.

Surprisingly, Costa showed that the first noise source – the dirty paper – has no effect on channel capacity. His proof of this relied on the use of a code book in which each message can be represented by a variety of alternative signals. In reference to his analogy, we refer to such a code book as a *dirty-paper code* [12][2].

In using a dirty-paper code, $\mathcal{U}$, to transmit a message, m, the transmitter performs the following steps:

1. Identify a coset of the code book associated with the message, $\mathcal{U}_m \subset \mathcal{U}$.
2. Search through $\mathcal{U}_m$ to find the code signal, $\mathbf{u}$, that is closest to the signal, $\mathbf{s}$, which will be added by the first noise source.
3. Transmit $\mathbf{x} = f(\mathbf{u}, \mathbf{s})$, where $f(\cdot, \cdot)$ is a function that is analogous to informed embedding. In Costa's construction,

[2]Costa did not invent the concept of a dirty-paper code. However, his analogy provides us with a colorful name.

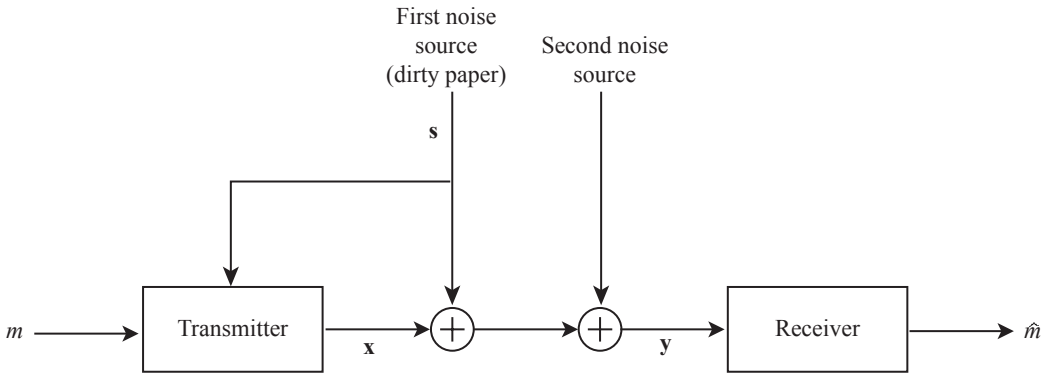

Fig. 9. "Dirty paper" channel studied by Costa.

$f(\mathbf{u}, \mathbf{s}) = \mathbf{u} - \alpha\mathbf{s}$, where α is a constant.

To decode a received signal, $\mathbf{y}$, using a dirty paper code, $\mathcal{U}$, the receiver performs the following steps:

1. Search the entire codebook for the closest code signal, $\hat{\mathbf{u}}$.
2. Identify the coset, $\mathcal{U}_{\hat{m}} \subset \mathcal{U}$, that contains $\hat{\mathbf{u}}$, and report reception of the message, $\hat{m}$, associated with that subset.

According to Moulin and O'Sullivan, [14], Costa's work was first brought to the attention of the watermarking community by Chen, who realized that the cover image can be considered to be a noise source that is perfectly known to the watermark encoder (dirty paper). The implication of Costa's analysis to watermarking is substantial – it implies that the channel capacity of a watermarking system is independent of the cover image. More recently, Moulin and O'Sullivan [14], [15] extended Costa's analysis to more realistic models of watermarking.

B. Practical dirty-paper codes

Unfortunately, Costa's research does not provide a practical solution to designing a dirty-paper code. His work was based on the use of random codes, and did not address the practical problem of efficient search. With random dirty-paper codes and exhaustive search, it is only possible to implement watermarks with very limited payloads (see, for example, the system studied in [12]). Thus, it is neccessary to introduce a structured code that allows for more efficient searches. A number of such codes have been proposed for watermarking.

Dither index modulation, proposed by Chen and Wornell [1], uses a lattice code, i.e. a code in which each code signal is a point on a regular L-dimensional lattice, where L is the dimension of the coding space. Different messages are represented with sub-lattices and finding the nearest code signal is simply a matter of quantization.

While most dither index modulation codes are very simple to implement, and allow for large payloads, they suffer some inherent problems. One problem is that rectilinear lattices are known to produce inefficient codes in high dimension, meaning that the payloads are not as large as possible for the robustness achieved. This can be solved by using more efficient lattices, such as can be found in [5], at the expense of complicating the implementation.

Another problem is that lattice codes are not robust to scaling in the coding space. This means that, if coding space is a linear projection of pixel space, image watermarks will not be robust to changes in contrast. Some non-linear coding spaces have been proposed [16], [3], but few experimental results with these spaces have been reported.

An alternative approach to designing structured codes is based on *syndrome coding*, proposed by [17] and applied to watermarking in [4]. This employs a modification of traditional error correcting codes (ECC's). In essence, syndrome codes encode messages in patterns of "errors" in coded bit sequences. Using a binary ECC, it is possible to arrange that many different bit sequences contain the same pattern of errors, thus allowing a given message to be encoded with a variety of different code words.

The central problem in using syndrome coding for watermarking is that syndrome codes do not introduce any redundancy, and, hence, do not provide any inherent robustness. The flipping of even a single bit in a syndrome-coded word is likely to change the pattern of errors, and thus change the message. This can be solved by combining a syndrome code with a lattice code. Messages are coded with syndrome codes first, and the resulting bit sequences are embedded using a lattice code. While such a system can have better performance than a lattice code alone, it still suffers from the basic problems with lattice codes.

C. Trellis based dirty-paper code

In this section, we propose a simple modification of a trellis code to produce a dirty-paper code. This code is likely to yield performance different from the types of codes discussed above, but we have yet to identify all the differences. Our principal motivation in developing the present code is that it allows a straightforward application of the informed embedding method described in Section II.

Figure 2 shows an example of a traditional trellis code. In this code, two arcs exit from each state: one bold arc that corresponds to a 1 bit in the coded message, and one non-bold arc corresponding to a 0 bit. This traditional trellis coding scheme assigns one unique path to each message once a starting state has been chosen.

To create a dirty paper code, the trellis is modified so that multiple alternative codewords can be obtained for each message. The basic idea is to have more than two arcs enter and exit each state, but still use each step of the trellis to encode a single bit. This is shown in Figure 10.

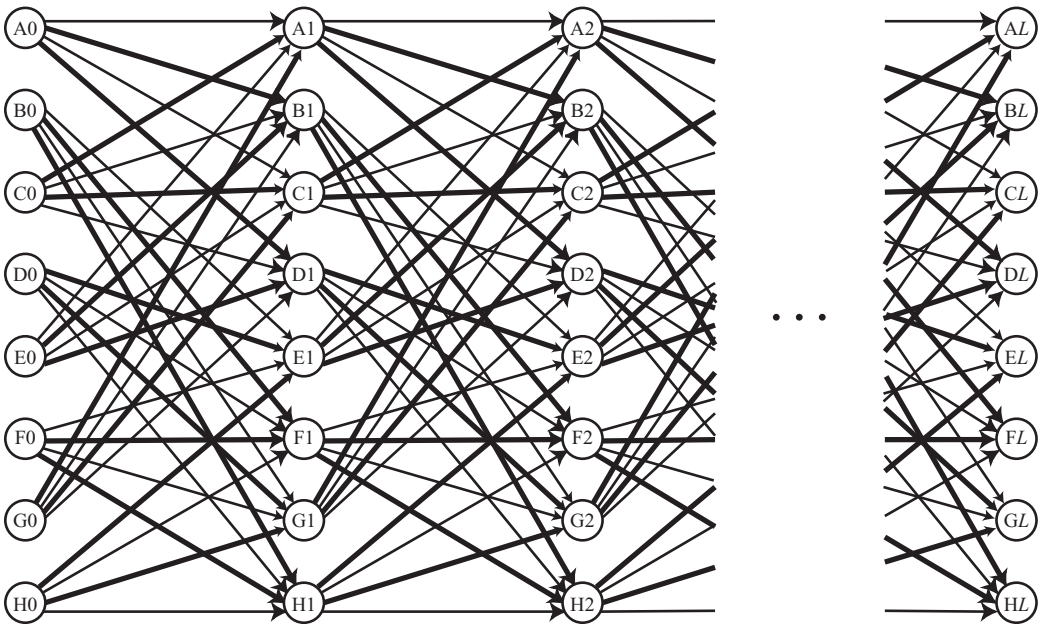

Fig. 10. Dirty-paper trellis with 8 states and 4 arcs per state (thus $S = 8$ and $A = 8 \times 4 = 32$). Each path from one of the nodes at the left (A0 through H0) to one of the nodes at the right (AL throught HL) represents a message. Bold arcs along a path correspond to 1 bits, and non-bold arcs correspond to 0 bits. Not that, since there are two bold and two non-bold arcs eminating from each node, and a message path may begin at *any* of the nodes at the left, a given bit sequence can be represented by a number of different paths.

Let us assume that we have A arcs and S states. Since there is no reason to privilege one state more than another, A/S arcs exit and enter each state. Half of those arcs will encode a 0 (non-bold arcs) and the other half will encode a 1 (bold arcs). There are many alternative paths in the trellis which encode the same message. Suppose we wish to encode an L-bit long message, m. If we do not impose any starting state, the number of codewords, n, which encode

the message m is given by:

$$n = S\left(\frac{A}{2S}\right)^L \tag{5}$$

We must now define how the embedder selects a path from the set of paths through the trellis that all represent that message that is to be embedded. Conceptually, this can be thought of as being done in two steps. First, we modify the trellis to eliminate all paths that *do not* encode the desired message. This is a simple matter of removing bold arcs from steps that should encode 0's, and removing non-bold arcs from steps that should encode 1's. In the resulting trellis, *every* possible path represents the desired message. An example of such a modified trellis is shown in Figure 11.

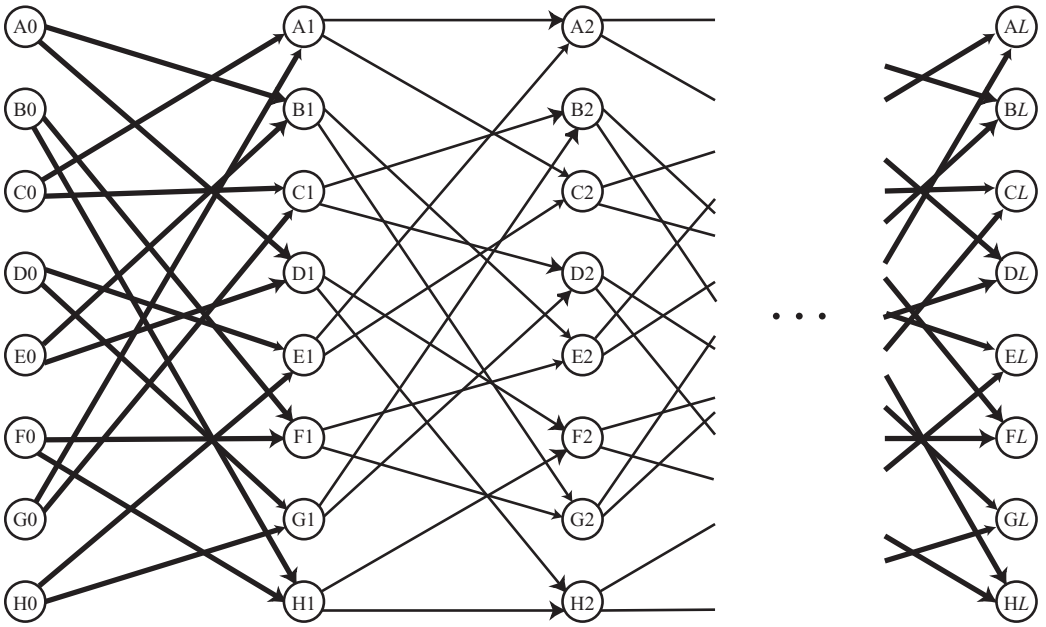

Fig. 11. Version of the dirty-paper trellis in Figure 10 modified to represent a message beginning with 100 and ending with 1. The first step in this trellis, from states A0 ... H0 to states A1 ... H1, has been modified by removing all the non-bold arcs, so every path represents a message that begins with 1. The second step, from A1 ... H1 to A2 ... H2, has had all the bold arcs removed, so the second bit in every path is a 0. And so on to the last step.

Second, the embedder applies the detection algorithm to the image, as described in Section II-A, except that it uses the modified trellis instead of the complete trellis. That is, it extracts a vector from the image, and then uses a Viterbi decoder to find the path through the modified trellis that yields the highest correlation with that extracted vector.

Once the highest-correlation path through the modified trellis has been identified, we can use the informed embedding algorithm of Section II-D to embed the watermark. Alternatively, for purposes of comparison, we can embed the watermark pattern by blindly adding it to the extracted vector.

During the detection process, the decoder applies the Viterbi algorithm to the *entire* trellis. This identifies the path that yields the highest correlation with the watermark. The hidden message is then decoded by looking at the bits represented by the arcs in that path.

D. Trellis structure

Given the general framework of the algorithm, we now proceed to investigate how the structure of the trellis affects performance. In particular, we examine how the number of arcs, A, and states, S, impact the effectiveness of the embedding method.

- If the number of arcs per state is greater than twice the number of states ($A/S > 2S$), there will be some *parallel* arcs

in the trellis, i.e for each bit value, there will be several arcs linking the same pair of states. In the extreme case of only a single state, ($S = 1$), all the arcs are parallel arcs as depicted in Figure 12.

- If the number of arcs per state is equal to twice the number of states ($A/S = 2S$), the trellis is fully connected i.e. for each bit value, each state is connected exactly once with itself and every other state.
- If the number of arcs per state is lower than twice the number of states ($A/S < 2S$), not all the states can be reached from any given state. This is the case depicted in Figure 10 ($S = 8, A = 32$).

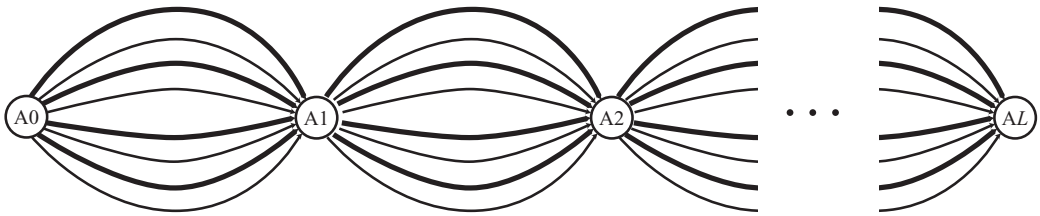

Fig. 12. Extreme case where there is only one state in the dirty paper trellis.

Two experiments were performed to investigate how the structure of the trellis influences the effectiveness of the watermarking scheme. In both experiments, we used uniformly distributed, random vectors as simulated extracted vectors. We represented each arc of the trellis with a length $N = 64$ vector[3]. Since we wished to examine the effectiveness of informed coding only, the mark output by the coder was blindly embedded with varying strengths, α. Immediately after embedding, the detector was applied to decode the watermark, and the resulting bit error rate (BER) was measured.

It must be noted that the bit error rate, as opposed to the message error rate, is effected in several different ways by the structure of the trellis. In particular, when the number of arcs per state is lower than the number of states ($A/S < S$), then whenever an error occurs it may take several steps in the trellis before we return to the correct path. This will introduce burst errors in the decoded messages, *increasing* the bit error rate. However, since multiple errors should reduce the correlation with the extracted vector, we expect them to happen rarely. In contrast, in a trellis with only one state, whenever an error occurs the decoder can immediately return to the correct path at the next iteration. Thus, in this configuration, a single error does not induce consecutive errors. However, the cost of single errors is less than for burst errors and they may therefore occur more frequently.

In the first experiment, we restricted the number of states to 1 and varied the number of arcs. According to Equation 5, this means that the number of codewords representing the same message is varying. The results are shown in Figure 13. One can observe a very significant reduction in bit error rate as the number of arcs increases from 2 to 64. Performance continues to improve as the number of arcs increases beyond 64, but the improvement is less dramatic. Since computational cost increases with the number of arcs, a good compromise appears to be $A = 64$. For 1 state and 64 arcs, Equation 5 yields the number of codewords, n, that encode a message.

In a second experiment, the number of states and the number of arcs were varied in such a way that the number of codewords representing the same message was kept constant at $n = 10^{2077}$. This means that A/S is held roughly constant at 64. The results are shown in Figure 14. Once again, the error rate quickly drops as the number of states grows before flattening as the number of states exceeds 64. Thus, there is little point in increasing the number of states beyond this point. The two experiments suggest that a configuration of 64 states and 64 arcs per states is a reasonable compromise.

[3]These experiments were performed before we decided to use only $N = 12$ length vectors for marking real images.

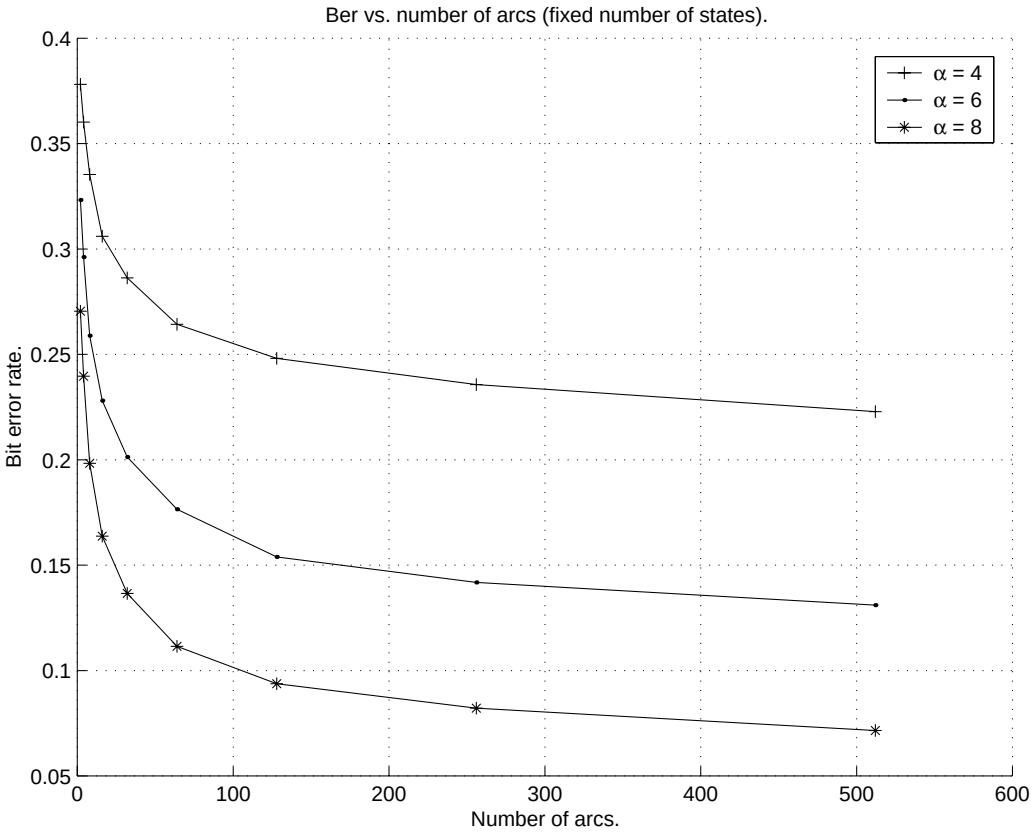

Fig. 13. Bit error rate versus number of codewords with one state trellis.

It should be noted that these experiments are quite preliminary. There are a wide variety of trellis configurations that may provide better results, and it will be interesting to study their behavior. In particular, it is interesting to examine the effect of different values of A/S. As A/S decreases, the minimum difference between paths for different messages increases, and the code becomes more robust against errors. As A/S increases, the number of different paths coding the same message increases, and the code becomes easier to embed.

E. Performance of informed coding and informed coding with informed embedding.

To demonstrate the improvement due to informed coding, 2000 images were watermarked with informed coding (using a trellis of 64 states and 64 arcs per state) and informed embedding. Immediately after embedding, the 2000 images were sent to a watermark detector and the message error rate (MER) was calculated. These results were then compared with the results from Section II-E in which *blind* coding and informed embedding were applied. We expect informed coding to improve the image fidelity, because the code offers the embedder a choice between a large number of signals to embed. The embedder chooses the one that is already most similar to the image, and hence will lead to the smallest fidelity impact.

Indeed, the image fidelity is significantly improved with informed coding. To quantify this, we calculated the average perceptual distance according to Watson's model in both cases. The results are summarized in Table II in which we see that the message error rate has been reduced from 12.5% to zero while simultaneously improving the image quality. The average perceptual distance using informed coding is about half that using blind coding.

To examine how much of this performance improvement was due to informed coding alone, we performed a second experiment in which 2000 images were watermarked using informed coding and *blind* embedding. The blind embedding

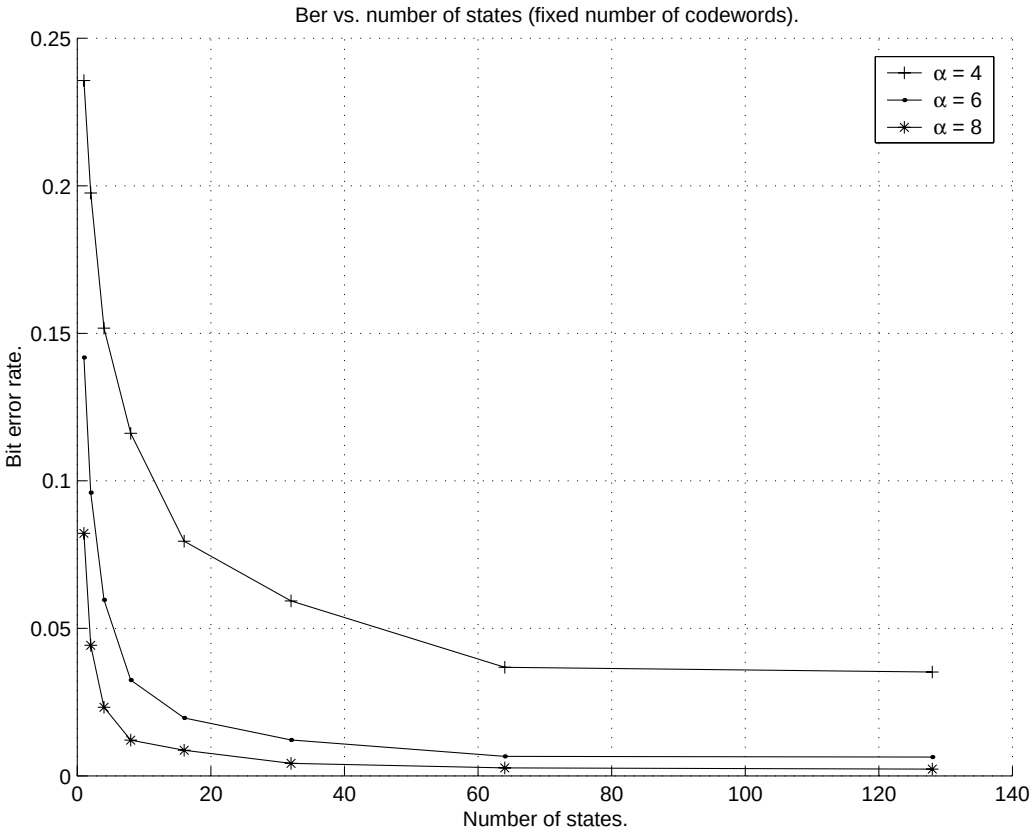

Fig. 14. Bit error rate with a constant number of codewords.

TABLE II
BLIND CODING VERSUS INFORMED CODING.

	Watson distance	**MER**
Blind coding + blind embedding	200.04	99.85%
Blind coding + informed embedding	201.06	12.5%
Informed coding + informed embedding	101.52	0%

strength was chosen to yield an average Watson distance of 101, i.e. roughly the same as the previous experiment with informed coding and informed embedding. Once again, the effectiveness of the embedder was measured. These results are summarized in Table III. This shows that our informed coding algorithm alone makes a significant improvement over blind coding, but its effectiveness is not satisfactory without informed embedding.

TABLE III
BLIND EMBEDDING VERSUS INFORMED EMBEDDING WITH INFORMED CODING.

	Watson distance	**MER**
Informed coding + blind embedding	101.79	56.55%
Informed coding + informed embedding	101.52	0%

IV. PERCEPTUAL SHAPING

At this point, the combination of informed coding and informed embedding is promising. However, many of the resulting watermarked images still have unacceptable fidelity. For example, Figure 15 illustrates how an image can be

distorted by our watermarking scheme. Artifacts are particularly noticeable in the region of the sky, where a noisy pattern can be detected. To alleviate this problem, we introduce a perceptual shaping stage to the proposed algorithm, based on Watson's perceptual measure [23].

Fig. 15. Image watermarked with informed coding and informed embedding.

The perceptual shaping is based on the E-PERC-OPT algorithm described in [7]. The basic idea is to shape the difference pattern $\mathbf{d}$ used in step 4 of the informed embedding algorithm (Section II-C). Each element of a watermark vector in our system is a single coefficient from the 8×8 block DCT of an image. Watson's model assigns a perceptual *slack* to each such coefficient, which indicates how much that coefficient may be changed before becoming perceptually noticeable. We can arrange the slacks for the low-frequency AC terms (Figure 3) into a vector, $\mathbf{s}$, such that the i'th component, $\mathbf{s}[i]$, is the slack for the i'th component of the extracted vector. The perceptual shaping of $\mathbf{d}$ is then performed as

$$\mathbf{d}'[i] = (\mathbf{d}[i]\mathbf{s}[i]^4)^{\frac{1}{3}} \tag{6}$$

This results in the vector, $\mathbf{d}'[i]$, that yields maximum correlation with $\mathbf{d}[i]$ for a given perceptual distance (see [7] for the derivation of this function).

In step 4 of the informed embedding algorithm, where we modify $\mathbf{c_w}$ to ensure that $R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b}) \geq R_t$, we no longer use Equation 4. Rather, we now modify $\mathbf{c_w}$ as follows:

$$\begin{aligned} \mathbf{d} &= \frac{\mathbf{g} - \mathbf{b}}{|\mathbf{g} - \mathbf{b}|} \\ \mathbf{d}' &= \text{ perceptually shaped version of } \mathbf{d} \\ \alpha &= \frac{R_t - R_0(\mathbf{c_w}, \mathbf{g}, \mathbf{b})}{\mathbf{d}' \cdot \mathbf{d}} \\ \mathbf{c_w} &\leftarrow \mathbf{c_w} + \alpha \mathbf{d}' \end{aligned} \tag{7}$$

This modification of the algorithm is not expected to effect the performance of the watermarking scheme since robustness is inherently ensured by the informed embedding algorithm.

To evaluate the effect of this perceptual shaping, 2000 images were watermarked with informed coding (64 states and 64 arcs per state), informed embedding and perceptual shaping. The Watson distance between the original and the watermarked images was then computed. A watermark detector was immediately applied to the 2000 watermarked images and the message error rate was computed.

The results are summarized in Table IV. While the message error rate has increased only insignificantly, the perceptual distance between watermarked and unwatermarked images has been reduced three-fold. The improvement in fidelity is easily seen in Figure 16 where the Watson-based perceptual shaping has been used. In comparison with Figure 15, one can noticed that the sky is far less distorted after incorporating perceptual shaping.

However, Figure 16 also illustrates a limitation with Watson's model. Some undesirable blocking artifacts have appeared along sharp edges, particularly on the right side of the image, where there is a thin black border. This is due to the fact that Watson's model is block-based and each block is independently examined. Blocks that contain sharp edges contain energy in all frequencies, and are erroneously judged to contain much texture that can mask watermark patterns.

TABLE IV
Effect of perceptual shaping.

	Watson distance	**MER**
Informed coding + informed embedding	101.52	0%
Informed coding + informed embedding + perceptual shaping	31.6	0.35%

Fig. 16. Fidelity improvement with the introduction of perceptual shaping.

V. Robustness experiments

Our previous experiments have examined only the embedding effectiveness of the watermark embedder, i.e. the performance when the watermarked image is not distorted between the times of embedding and detection. In practice, watermarked content will be subjected to a variety of distortions before reaching the detector. Watermarks designed to survive legitimate and everyday usage of content, e.g. low pass filtering, noise, JPEG compression, are referred to as *robust* watermarks.

In this section, we measure the effect of a wide range of distortions on three different watermark algorithms: (i) blind coding, informed embedding, no shaping (bciens), (ii) informed coding, informed embedding, no shaping (iciens), and (iii) informed coding, informed embedding and shaping (icies). Algorithms employing blind embedding were not tested, since they have been found to have unacceptable performance even without image distortions. We report robustness results for addition of Gaussian noise, low pass filtering, valumetric scaling, and JPEG compression. For each class of distortion, the 2000 watermarked images were modified with a varying magnitude of distortion. The message error rate was then computed. Following the practice suggested in [10], we considered the watermarking scheme to be robust if at least 80% of the watermarks were correctly retrieved, i.e. the message error rate is below 20%.

A. Gaussian noise

Normally distributed noise with mean 0 and standard deviation σ was added to each of the watermarked images. The experiment was repeated for different standard deviations, σ, and the message error rate has been computed.

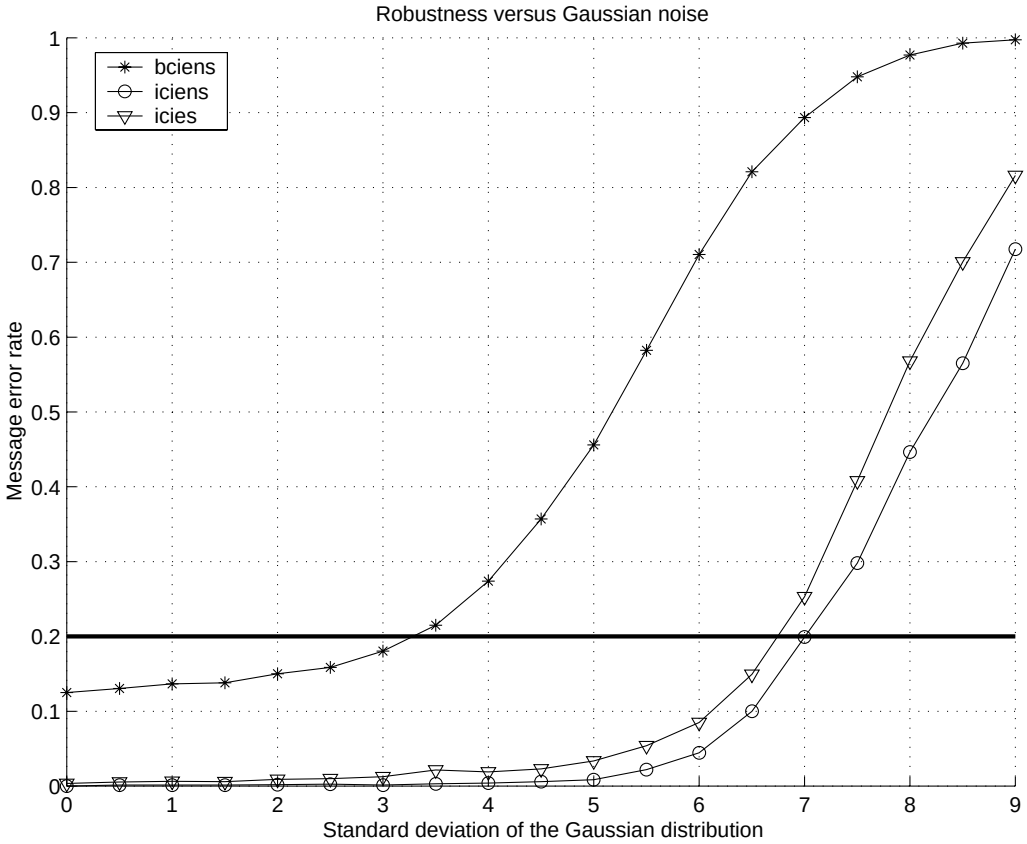

Fig. 17. Robustness versus Gaussian noise.

The results are summarized in Figure 17. Notice that the two schemes that use informed coding – iciens (no perceptual

shaping) and icies (with perceptual shaping) – have quite similar performance. This is because the perceptual shaping does not interfere with informed embedding, as explained in Section IV. Remember, however, that images embedded using the icies algorithm have substantially better fidelity.

The gain obtained by introducing informed coding is quite noticeable. If we establish a threshold at a message error rate of 20%, then blind coding with informed embedding and no shaping (bciens) is robust against additive Gaussian noise with standard deviation, σ, of only 3.25 or less. In contrast, the two informed coding methods are robust to standard deviations up to 6.5.

Fig. 18. Watermarked reference image subjected to a Gaussian noise with standard deviation $\sigma = 6.5$.

Figure 18 shows the watermarked image of Figure 16 after being altered by additive Gaussian noise with standard deviation $\sigma = 6.5$. The degradation is clearly observable in uniform areas like the sky.

B. Low pass filtering

All three watermarking schemes use only low frequency DCT coefficients. As a result we expect them to be quite resilient against low pass filtering. To verify this, the watermarked images were filtered with Gaussian filters of width σ_g. The experiment was repeated for different values of σ_g and the message error rate computed.

Figure 19 summarizes the results. Once again, the informed coding methods have similar performance and the improvement between blind coding and informed coding is clear. With blind coding, the message error rate reaches 20% with a Gaussian filter of width $\sigma_g = 0.7$. In contrast, with informed coding, the images can be filtered with a Gaussian filter of width $\sigma_g = 1.5$ before the same message error rate is reached.

Figure 20 shows the watermarked image shown in Figure 16 after being filtered by a Gaussian filter of width $\sigma_g = 1.5$. Clearly, the image fidelity is very poor.

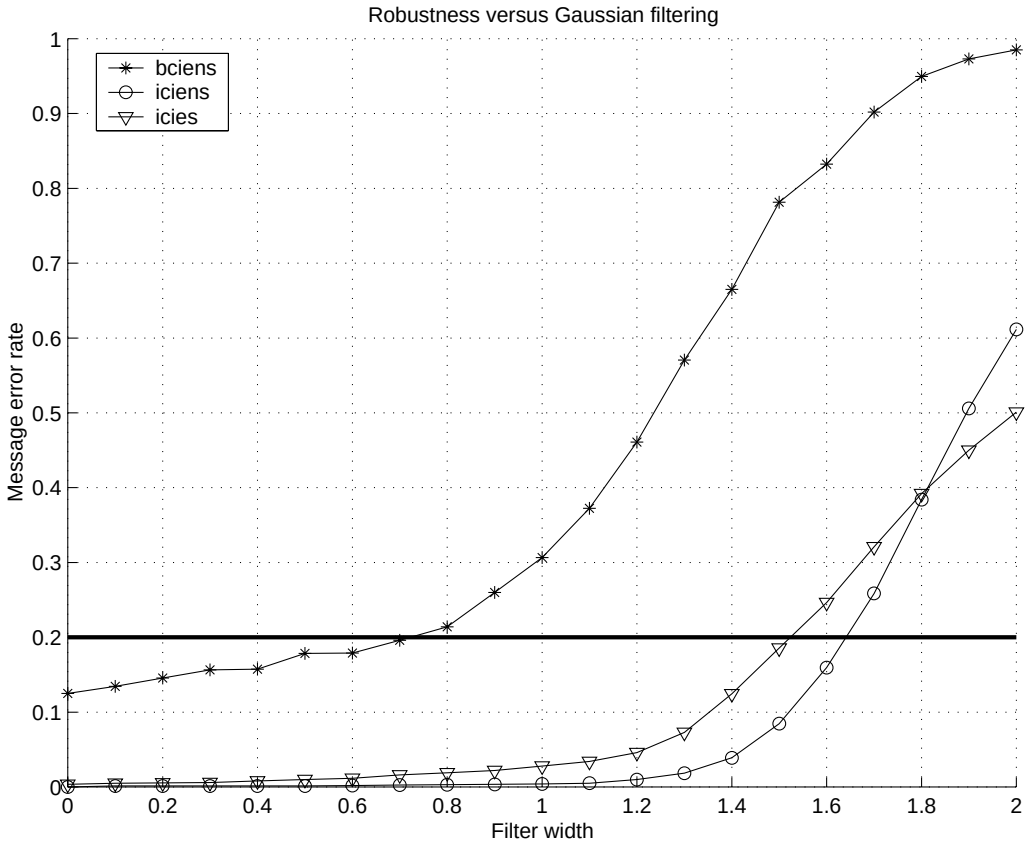

Fig. 19. Robustness versus low pass filtering.

C. Valumetric scaling

Another simple, but important distortion is changing amplitude. That is

$$\mathbf{c_n} = \nu \mathbf{c} \tag{8}$$

where $\mathbf{c}$ is an image and ν a scaling factor. This corresponds to a change of brightness and contrast for images and video. This attack is of particular interest for us, and was indeed the main weakness of the watermarking schemes proposed by Chen and Wornell [1].

Two tests were performed. The first reduced the image intensities, i.e. ν varied from 1 to 0.1. The second experiment increased the image intensities as ν increased from 1 to 2.

The results of the first test are summarized in Figure 21. As usual, the two informed coding methods have very similar performance and are superior to the blind coding method. The results clearly demonstrate that our watermarking schemes are resilient against valumetric scaling down. Whatever watermarking scheme is chosen, its performance remains the same down to a scaling factor of 0.1.

A scaling with a factor of 0.1 produced a serious image degradation. An example of a scaled image cannot be shown in this article because, with such a scaling factor, the image is almost completely black. However, even if the distorted image is much darker, the hidden message is still correctly extracted. This is because valumetric scaling multiplies all the correlation scores by the same scaling factor (modulo some rounding). As a result the best path along the trellis remains the same.

In a second test, we investigated the robustness of our watermarking methods to valumetric scaling for $1 \leq \nu \leq$

Fig. 20. Watermarked reference image low pass filtered with a Gaussian filter of width $\sigma_g = 1.5$.

2. Figure 22 summarizes the results. Once more, the two informed coding methods have similar performance and significantly out-perform the blind coding method. While the blind coding method survives scaling by a factor of $\nu \approx 1.1$, the two informed coding methods remain robust for scale factors up to $\nu = 1.3$.

The robustness to scaling intensities up is much worse than the robustness to scaling intensities down. This is because, in addition to rounding, valumetric scaling up introduces some clipping i.e. all the pixel values above 255 after scaling are truncated to 255. This has a more severe impact than rounding. Figure 23 shows the watermarked image of Figure 16 after valumetric scaling up with a factor $\nu = 1.3$. The image is globally brighter, but one can also notice that some textured areas have become uniformly white after scaling (the flowers in the central bottom part).

D. Lossy compression

Here we tested the effects of JPEG compression. The JPEG standard does not actually specify a compression method, so different encoders can, in principle, have different effects on watermarks. However, there is a conventional method of encoding which is very common. This specifies the quantization values for DCT coefficients by multiplying a quantization matrix (Table V) by a global quantization level, Q. For example, if $Q = 2$, the DC term is quantized with a quantization level of $q = 32$ and the highest-frequency term with $q = 198$. The quantization level is related to a user-specified *quality factor*, QF, in the range of 0 to 100, as [11]

$$Q = \begin{cases} 50/QF & \text{if } QF < 50 \\ 2 - 0.02 \cdot QF & \text{if } QF \geq 50 \end{cases} . \tag{9}$$

To test the effects of JPEG, then, it is sufficient to apply only this quantization. This is preferable to using some encoder from a third party, as we cannot be sure of the algorithms used (and the algorithms may change over time, making it difficult to reproduce results).

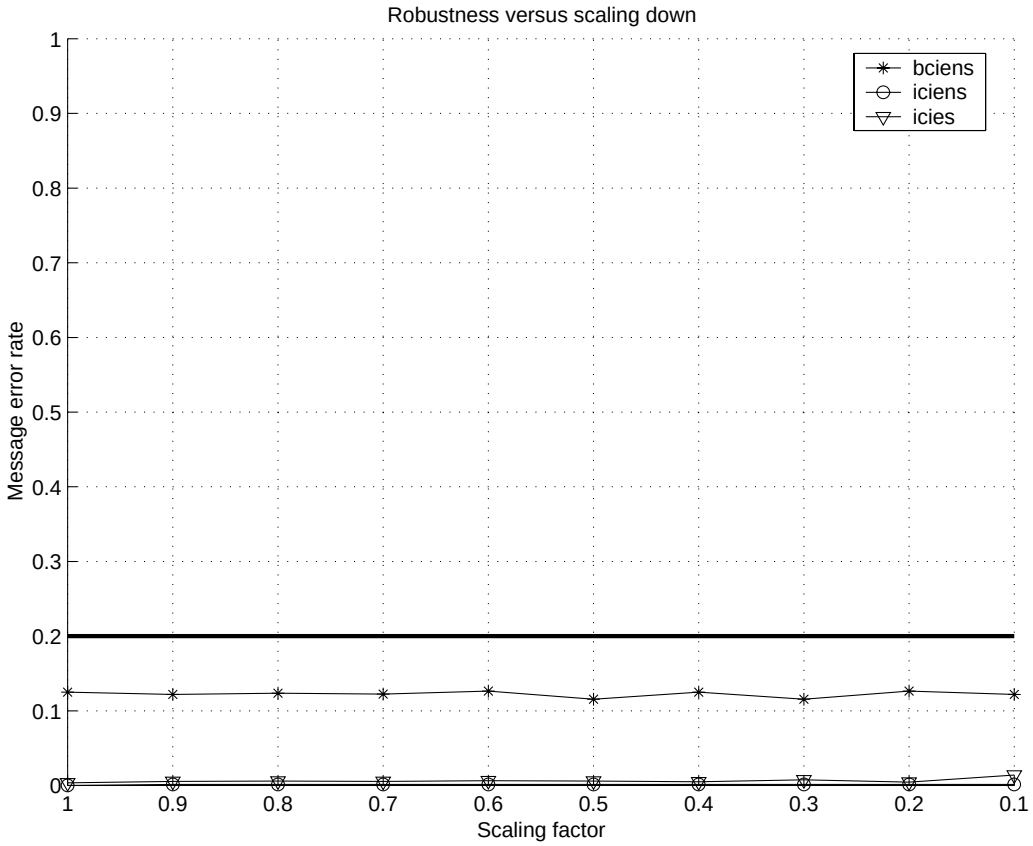

Fig. 21. Robustness versus valumetric scaling down.

The effects of JPEG compression were simulated by applying quantization in the DCT domain. The block-DCT transform was computed for each watermarked image. The DCT coefficients were then quantized according to the following Equation

$$\mathbf{c_n}[i] = q \left\lfloor \frac{\mathbf{c}[i]}{q} + 0.5 \right\rfloor \tag{10}$$

where q is the quantization value obtained as described above. After quantization, the inverse block-DCT was applied. The watermarked images were quantized with different values for the parameter Q and the message error rate was computed.

TABLE V
LUMINANCE QUANTIZATION MATRIX USED IN JPEG.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
48	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

The results are summarized in Figure 24 and expressed in terms of JPEG quality factor QF. The reader should keep in mind that it is easy to switch between the JPEG quality factor QF and the global quantization level Q thanks to

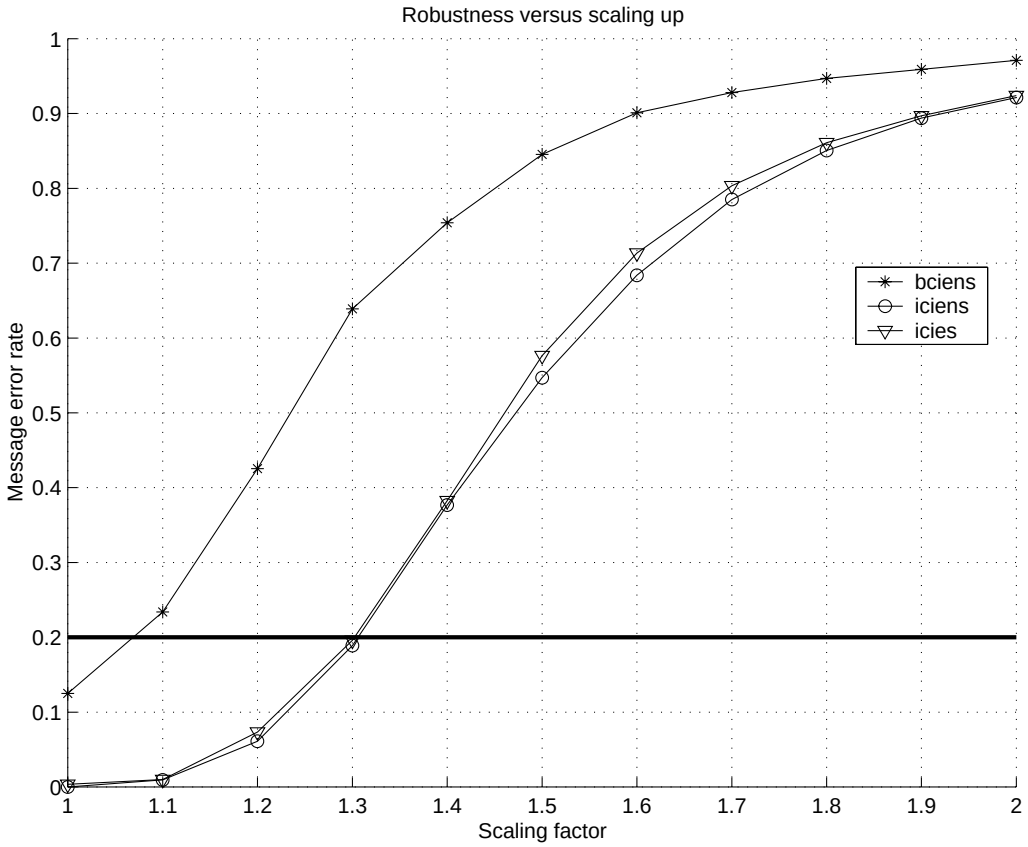

Fig. 22. Robustness versus scaling up.

Equation 9. Again the two informed coding methods have similar performance and significantly out-perform the blind coding method. The blind encoding method only remains robust for values of QF less than 65, while the informed coding methods remain robust for $QF \leq 33$.

Figure 25 shows the impact of lossy compression with a quality factor $QF = 33$. The image fidelity is very degraded at this point.

VI. Conclusion and future work

Five different watermarking schemes have been investigated in this paper and their performances have been gathered in Table VI. On one hand, informed coding reduces the fidelity impact of the watermarking process and, on the other

TABLE VI
Embedding performance with five schemes

	Watson distance	**MER**
Blind coding + blind embedding	200.04	99.85%
Blind coding + informed embedding	201.06	12.5%
Informed coding + blind embedding	101.79	56.55%
Informed coding + informed embedding	101.52	0%
Informed coding + informed embedding + perceptual shaping	31.6	0.35%

hand, informed embedding ensures a specified robustness. However, only the combination of those two approaches gives satisfactory results. Furthermore, the fidelity impact of those algorithms can be decreased by introducing a perceptual

Fig. 23. Watermarked reference image scaled with a scaling factor $\nu = 1.3$.

model. The final watermarking scheme (informed coding + informed embedding + perceptual shaping) exhibits good robustness against a wide range of everyday distortions.

Possible areas of further research include:

• *Increasing the capacity* The watermarking schemes presented here embed one bit for every 64 pixels in an image. With a 240×368 image, 1380 bits have been embedded. However some researchers are still interested in increasing the capacity of the watermark beyond this point. One possible solution would be to associate more than one bit to the different arcs of the trellis. Some preliminary theoretical and mathematical analysis has been conducted and the results have been checked with some experiments. They show that it might be possible to increase the capacity of our watermarking scheme with this trick, but it will cost more embedding strength, i.e. the perceptual impact induced by the watermarking process will be greater.

• *Better perceptual models* It has already been noted in Section IV that Watson's perceptual model has introduced an annoying blocking artifact. This is inherent to Watson's model, because the perceptual slacks take in account only the pixels in a given DCT block. Smoothness at the edges between the DCT blocks is consequently not guarantied. We are currently studying the application of a model due to Christian J. van de Branden Lambrecht and Joyce E. Farrell [20], which is based on Gabor filters and does not divide images into blocks. Alternatively, we might employ pixel-based models such as that proposed by Voloshinovsky [22].

References

[1] B. Chen and G. W. Wornell. Digital watermarking and information embedding using dither modulation. In *IEEE Second Workshop on Multimedia Signal Processing*, pages 273–278, 1998.

[2] B. Chen and G. W. Wornell. An information-theoretic approach to the design of robust digital watermarking systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1999.

[3] B. Chen and G. W. Wornell. Preprocessed and postprocessed quantization index modulation methods for digital watermarking. In *Security and Watermarking of Multimedia Contents II*, volume SPIE-3971, pages 48–59, 2000.

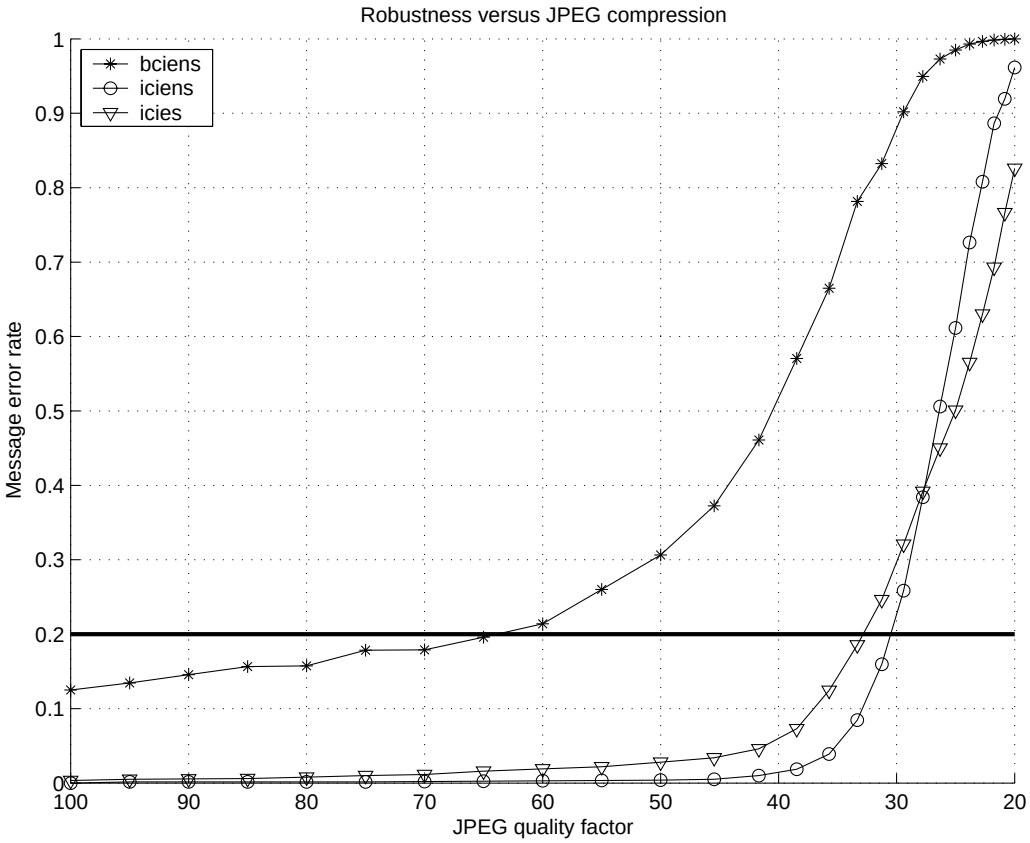

Fig. 24. Robustness versus JPEG quantization.

[4] Jim Chou, S. Sandeep Pradhan, and Kannan Ramchandran. On the duality between distributed source coding and data hiding. *Thirty-third Asilomar conference on signals, systems, and computers*, 2:1503–1507, 1999.

[5] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices, and groups*. Springer-Verlag, 1988.

[6] M. Costa. Writing on dirty paper. *IEEE Trans. Inform. Theory*, 29:439–441, 1983.

[7] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2001.

[8] I. J. Cox, M. L. Miller, and A. McKellips. Watermarking as communications with side information. *Proc. IEEE*, 87(7):1127–1141, 1999.

[9] J. J. Eggers, J. K. Su, and B. Girod. A blind watermarking scheme based on structured codebooks. In *IEE Seminar on Secure Images and Image Authetication*, pages 4/1–4/21, 2000.

[10] M. Kutter and F. A. P. Petitcolas. A fair benchmark for image watermarking systems. *Security and Watermarking of Multimedia Contents*, Proc. SPIE - 3657:226–239, 1999.

[11] C-Y Lin and S-F Chang. Semi-fragile watermarking for authenticating JPEG visual content. In *Proc of SPIE, Security and Watermarking in Multimedia Contents II*, volume 3971, 2000.

[12] M. L. Miller. Watermarking with dirty-paper codes. 2:538–541, 2001.

[13] M. L. Miller, I. J. Cox, and J. A. Bloom. Informed embedding: Exploiting image and detector information during watermark insertion. In *IEEE International Conference on Image Processing*, September 2000.

[14] P. Moulin and J. A. O'Sullivan. Information-theoretic analysis of information hiding. *preprint*, available from http://www.ifp.uiuc.edu/˜moulin/paper.html, 1999.

[15] P. Moulin and J. A. O'Sullivan. Information-theoretic analysis of information hiding. In *Proc. ICASSP'00*, 2000.

[16] R. Petrovic, J. M. Winograd, K. Jemili, and E. Metois. Data hiding within audio signals. In *Fourth International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, 1999.

[17] S. S. Pradhan and K. Ramchandran. Distributed source coding: symmetric rates and applications to sensor networks. In *Proc. IEEE Data Compression Conference*, pages 363 –372, 2000.

[18] M. Ramkumar. *Data hiding in multimedia: Theory and applications*. PhD thesis, New Jersey Institute of Technology, 1999.

[19] C. E. Shannon. Channels with side information at the transmitter. *IBM Journal of Research and Development*, pages 289–293, 1958.

[20] Christian J. van den Branden Lambrecht and Joyce E. Farrell. Perceptual quality metric for digitally coded color images. *Proc. EUSIPCO*, pages 1175–1178, 1996.

[21] A. J. Viterbi. *CDMA: principles of spread spectrum communications*. Addison Wesley Longman Inc., 1995.

[22] S. Voloshynovskiy, A. Herrigel, N. Baumgaetner, and T. Pun. A stochastic approach to content adaptive digital image watermarking. In *Third International Workshop on Information Hiding*, 1999.

[23] Andrew B. Watson. DCT quantization matrices optimized for individual images. *Human Vision, Visual Processing, and Digital Display IV*, SPIE-1913:202–216, 1993.

Fig. 25. Close up detail of the watermarked reference image compressed with a quality factor equal to 33%.