Facilitating watermark insertion by preprocessing media

Ingemar J. Cox, Matt L. Miller
NEC Research Institute
4 Independence Way
Princeton
NJ 08540

March 6, 2004

Abstract

There are several watermarking applications that require the deployment of a very large number of watermark embedders. These applications often have severe budgetary constraints that limit the computation resources that are available. Under these circumstances, only simple embedding algorithms can be deployed, which have limited performance. In order to improve performance, we propose preprocessing the original media. It is envisaged that this preprocessing occurs during content creation and has no budgetary or computational constraints. Preprocessing combined with simple embedding creates a watermarked Work, the performance of which exceeds that of simple embedding alone. However, this performance improvement is obtained without any increase in the computational complexity of the embedder. Rather, the additional computational burden is shifted to the preprocessing stage. A simple example of this procedure is described and experimental results confirm our assertions.

1 Introduction

There are a number of applications of watermarking in which it is necessary to deploy a very large number of watermark embedders. In such situations, economic constraints are often severe and constrain the computational resources that are available for embedding. Unfortunately, high

performance - as measured by effectiveness, fidelity and robustness - watermark embedders commonly require very substantial computational resources, especially when perceptual modeling [22, 13], informed coding [15, 3, 5][1] and/or informed embedding [20] are utilized.

We address this dilemma by proposing a two stage procedure in which a substantial fraction of the computational workload is performed as a preprocessing step on the media prior to its release to the general public. This preprocessing step is designed to permit, at a later time, subsequent watermark embedding based on computationally simple algorithms that are very economic.

Our solution is appropriate in situations where content can be modified before it reaches the watermark embedders. Section 2 discusses two examples where this is common. The first example uses watermarks for transaction tracking (also known as *fingerprinting*) during consumer playback of copyrighted material. Here, each player embeds a unique watermark into everything it plays. The watermarks may be used to identify the source of any content that is subsequently distributed illegally. The second example uses watermarks to prevent certain forms of illegal copying. Here, a `copy_mark` is added to video as it is being recorded in a consumer device, differentiating the original from the copy. The `copy_mark` indicates that it is illegal to make a second-generation copy of the copy.

In Section 3, we describe the basic principles behind preprocessing and a two-step watermarking process. Some performance implications are discussed in Section 4. An illustrative implementation of preprocessing is then described and tested in Section 5. Finally, a discussion of results and future work are contained in Section 6.

[1] Note that the term "preprocessing" as used in [3] differs from our usage here.

2 Motivation

We are motivated by watermarking applications in which watermarks must be inexpensively embedded. Below, we describe two such applications, both for video: the DiVX transaction tracking system and the proposed Galaxy copy-protection system. Many aspects of these applications severely limit the power of the embedders that may be used. At the same time, both applications allow expensive preprocessing of video before it reaches the watermark embedders, making our solution possible. How this preprocessing can be used to improve the performance of inexpensive embedders is described in Section 3.

2.1 DiVX transaction-tracking system

In late 1996, the DiVX Corporation[2] released an enhanced DVD player based on a pay-per-play business model. DiVX disks used proprietary encryption, so they could only play in DiVX-enabled DVD players. The players communicated with the DiVX Corporation over the phone lines, allowing DiVX to monitor the number of times a given player played each disk, and bill the player's owner accordingly.

In order to allay the piracy concerns of Hollywood studios, DiVX implemented a number of security technologies. One of these was a watermark-based system for transaction tracking. Each DiVX player embedded a unique watermark in the analog NTSC video signal during playback of a movie. These transaction watermarks were intended to be used to track the source of any pirated video that originated from the DiVX customer base. As players were connected to the DiVX corporation by phone, this would make it possible to quickly identify the pirate.

The DiVX DVD player was a consumer level product and, as such, was extremely price

[2]The DiVX Corporation filed for bankrupcy about one year after their product launch.

sensitive. Accordingly, the computational resources allocated to embedding the transactional watermark had to be small. This limitation on computational resources was further exacerbated by the requirement that the watermarks be embedded in real time. There are no published details regarding the design of the watermark embedder deployed by DiVX, but a personal communication between one of the authors and a Hollywood executive suggests that the fidelity was poor. This is to be expected given the design constraints.

The solution proposed here would preprocess the video prior to release of the DVD disc in order to improve the performance of the watermark embedder. This preprocessing could have been performed during DiVX's proprietary media preparation.

2.2 Generational copy-control for DVD

In 1997, the Copy Protection Technical Working Group issued a request for proposals for a watermarking system to prevent illegal copying of DVD movies. The basic idea is that each DVD recorder will contain a watermark detector, and will refuse to record video that contains certain watermarks. They received eleven proposals. After several rounds of testing and negotiations, these were reduced to the *Millenium* system, proposed by Philips, DigiMarc, and Macrovision, and the *Galaxy* system, proposed by NEC, IBM, Sony, Hitachi, and Pioneer. Both these systems involved embedding watermarks in video to implement one of the more difficult requirements in the request for proposals, known as *generational copy-control* or *copy generation management*.

Copy generation management is intended to allow a single generation of copies to be made from a master, but no subsequent copies to be made from the first generation copies. The requirement arises because consumers in the US are permitted by law to record television broadcasts for viewing later. This right was accorded consumers after the introduction of the video cassette

recorder when Hollywood studios sued electronics manufacturers alleging that such devices enabled widespread piracy of movies [1]. DVD recorders are covered by this law, but the studios recognize that digital recording is a potentially greater threat than analog recording since there is no degradation in video quality with each generational copy.

In order to reduce the threat of piracy, content owners envisage labeling broadcasted material as `copy_once` and subsequently labeling the material as `copy_no_more` after recording. A number of technical solutions to copy generation management were proposed in the context of DVD recorders. These are discussed in [2, 17]. The solution proposed in the Galaxy system used a fixed watermark to encode the `copy_once` state, and add a second, *copy_mark*, to encode the `copy_no_more` state. This second watermark would be added during recording, within the consumer DVD recorder.

Because the second watermark embedder was to be incorporated into consumer devices, it was subject to severe economic constraints. These economic constraints mandated that the embedder circuitry not exceed 50K gates, which precluded the use of a framebuffer. A consumer DVD recorder is expected to have both analog and digital video input. In the analog case, e.g. NTSC, watermark embedding was required to proceed in real-time. The digital video input is assumed to be a compressed MPEG-2 stream. Copy_mark embedding must therefore occur in both the compressed and baseband video domains. Moreover, compressed and baseband watermarks must be completely compatible, i.e. a watermark embedded in the MPEG domain must be detectable in the baseband domain and vice versa.

Embedding into the MPEG-2 stream introduces several additional limitations. First, because there is no possibility of employing a frame buffer, the watermark must be embedded without full decompression. This may be accomplished by directly modifying the compressed video stream

in a manner that changes the underlying baseband video [10]. Second, MPEG-2 recording may occur *faster* than real-time, making it necessary to embed the watermark in up to eight times real-time. Third, to maintain the integrity of the transport stream, it is necessary to ensure that the size of individual transport packets remain unchanged by the watermarking process. An embedder that satisfies all these constraints is unlikely to be capable of performing the processing required to embed high-fidelity, robust watermarks.

In the Galaxy system, a primary component of our solution to the `copy_mark` embedding problem was the use of preprocessing[3]. At the time that the `copy_once` mark was embedded, video was also processed to ease the task of subsequent `copy_mark` embedding. The principles for performing this type of preprocessing are the subject of the remainder of this paper. For the sake of simplicity, we describe these principles in the context of systems using conventional, baseband embedders. However, they apply equally well to any embedding method that embeds weak watermarks into the baseband video, even if that embedding is performed by modifying the compressed stream.

3 Media preprocessing

One of the main difficulties with cheap watermark embedders is that their performance is highly dependent on the cover Works to which they are applied. An embedder might perform well on one Work, successfully embedding a high-fidelity, robust mark, while completely failing to embed in another Work. The idea of preprocessing is to modify all the Works beforehand, altering them such that an inexpensive embedder will perform well.

[3]Note that, although the basic principles presented in this paper were developed during our work on the Galaxy proposal, the actual algorithms presented are not those used in Galaxy.

We illustrate the idea of preprocessing by applying it to three basic watermarking systems: a simple, zero-bit[4] linear-correlation system (Section 3.1), a zero-bit, normalized-correlation system (Section 3.2), and a one-bit, normalized-correlation system (Section 3.3). Admittedly, these basic systems are quite rudimentary, and don't have the theoretical justification of more recent systems based on dirty-paper coding (see [6, 21] for some recent examples). Nevertheless, systems like these have long proven useful in practice, and they serve nicely as test-beds for the concept of media preprocessing. In principle, the ideas presented here should also be applicable to more sophisticated systems.

3.1 Preprocessing for a linear correlation system

In a zero-bit, linear-correlation watermarking system, the detector tests for the presence or absence of a watermark by computing the linear correlation between a received Work, $\mathbf{c}$, and a reference pattern, $\mathbf{w_r}$:

$$z_{\text{lc}} = \frac{1}{N}\mathbf{c} \cdot \mathbf{w_r} = \frac{1}{N}\sum_{i}^{N} \mathbf{c}[i]\mathbf{w_r}[i]. \tag{1}$$

If z_{lc} is greater than a detection threshold, τ_{lc}, then the detector reports that the watermark is present. The interested reader is directed to [9] for background on the justification and interpretation of this type of system.

The simplest method of embedding watermarks for such a system is with a *blind* embedder, in which the embedded pattern and embedding strength are independent of the cover Work. The structure of a blind embedder is shown in Figure 1. This contrasts with *informed* embedding, as shown in Figure 2, where the embedding strength can be adjusted to ensure that a watermark is successfully embedded in every cover Work.

[4] A system that can embed 2^n distinct watermark messages is said to embed an n-bit watermark. Thus, a *zero*-bit system can embed only $2^0 = 1$ possible message. The watermark is either present or absent.

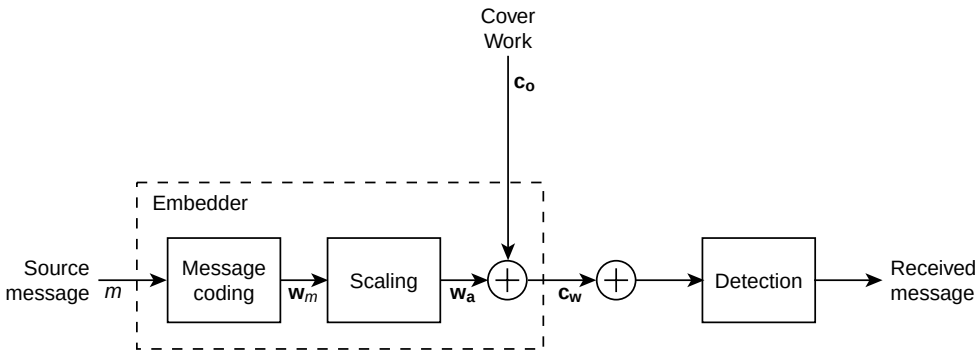


Figure 1: Watermarking using blind embedding

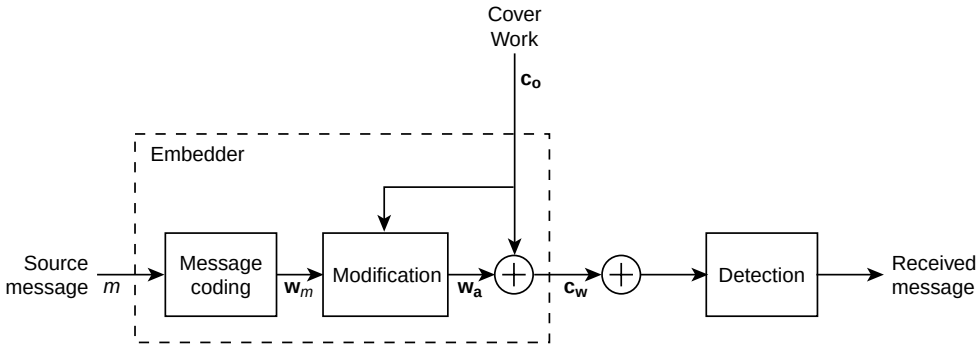


Figure 2: Watermarking using informed embedding

Blind embedding is computationally trivial. For example, a watermark can be added to a video stream (in baseband) without requiring that the frames be buffered. However, a blind embedder will necessarily fail to embed the watermark into some content, making its embedding effectiveness less than 100%. This makes it unacceptable for many applications in which the watermark *must* be embedded, even at the expense of occasional reductions in fidelity. An informed embedder, on the other hand, can guarantee 100% effectiveness by automatically adjusting the embedding strength (and hence the fidelity) for each cover Work, but to do so, it must examine the entire cover Work before embedding the mark, so a video system would require the expense of a frame buffer. Thus, informed embedding can be substantially more expensive than blind embedding. Below, we describe the two types of embedding in more detail, and then show how informed embedding can be split into a preprocessing step, followed by an inexpensive, blind embedder.

To understand the behavior of embedders, it is useful to consider a geometric model of the problem, in which cover Works are represented as points in a high dimensional marking space. In blind embedding a fixed vector that is independent of the cover Work is added to each Work, the intention being to move the cover Work into the detection region. A two-dimensional geometric model is illustrated in Figure 3a. If a simple correlation detector is used, then this detection region is a half-plane the boundary of which is denoted by the vertical line in Figure 3a. Unwatermarked cover Works lie to the left of this boundary and are denoted by the open circles. Notice that some cover Works are closer to the boundary than others[5]. The horizontal arrows represent the watermarking process which moves the cover Work towards, and hopefully into,

[5]In fact, it is also possible for an unwatermarked Work to be to the right of the boundary. This would denote a false positive.

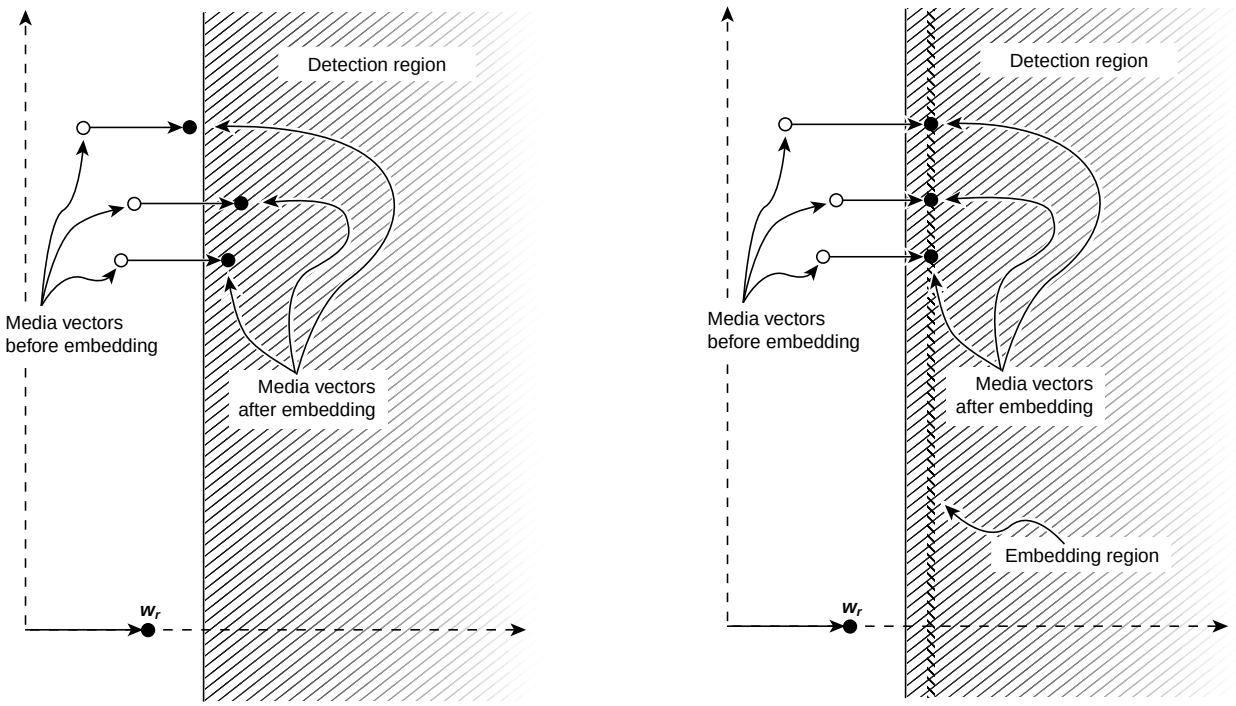
the detection region. This is also illustrated in Figure 3a where the majority of cover Works have indeed been moved into the detection region, but one cover Work has not. The embedder is said to have failed to watermark this particular cover Work, i.e. its effectiveness is less than 100%.

Clearly, if the magnitude of the arrows is larger, then more cover Works will be successfully watermarked. However, a compromise must be made between the strength of the watermark and the fidelity of the watermarked Work.

In contrast to blind embedding, informed embedding allows us to automatically vary the strength of the watermark based on the cover Work. Figure 3b illustrates the effect of an informed embedder in which a watermark of different magnitude is added to each cover Work, such that all watermarked Works are guaranteed to be a fixed distance within the detection region. Using such an informed embedder ensures that all watermarked Works will lie in the narrow shaded region of Figure 3b. We refer to this region as the *embedding* region.

It should be noted that the systems illustrated in Figure 3 are not strictly comparable because they solve subtly different problems. The blind embedder in Figure 3a is trying to embed the most robust watermark possible within a given prescribed limit on perceptual distortion. By contrast, the informed embedder in Figure 3b is trying to embed the least-perceptible watermark possible within a given prescribed limit on robustness. Thus, the blind embedder deals with the problem of *unwatermarkable content* – content which cannot be watermarked within a prescribed fidelity limit – by failing to embed, while the informed embedder deals with this problem by relaxing the fidelity constraint[6]. Which approach is better depends on the application. In some

[6]This problem sometimes arises with the type of simple watermarking systems we are discussing here. It can be reduced by employing dirty-paper codes [8, 19], which allow the embedder to embed any of a number of different patterns for each given message. In particular, dirty-paper codes based on lattice quantization [4, 15] can



(a) (b)

Figure 3: Geometric interpretation of two ways to embed marks for a linear-correlation detector: (a) blind embedding, with fixed visibility, (b) informed embedding, with fixed robustness. The empty circles denote unwatermarked Works and are randomly distributed in a high dimensional vector space. The vertical line denotes the detection boundary when a linear correlator is used. The x-axis is aligned with the watermark reference vector. In (a), addition of the reference vector to unwatermarked Works moves these Works to locations denoted by the solid circles which are usually, but not necessarily, within the detection region. This gives roughly constant fidelity at the expense of variable robustness (and occasional failure to embed). In (b), the reference vector is scaled to ensure that every watermarked Work lies a fixed distance inside the detection region, giving roughly constant robustness at the expense of variable fidelity.

applications, maintaining fidelity (as specified with some numerical measure) is more important than ensuring that every Work is marked. In others, the watermark is more important. We have argued elsewhere [11, 20] that the latter type of application is very common, and, for the remainder of this paper, we assume that this is the type of application in which our system will be employed. The difficulty we face is that informed embedding, because it requires a frame buffer, is too costly for our assumed application.

Now let us consider a two step process in which informed preprocessing is used to guarantee that subsequent blind embedding will be successful. Figure 4 shows how such a system might work. Here, the preprocessing stage modifies each original cover Work (open circles) so that the processed Works (grey circles) all lie within a narrow region close to, but outside of the detection region. We refer to this narrow region as the *prepping* region. Since the prepping region is outside the detection region, no watermarks are detected in the preprocessed content. However, when a simple blind embedder is subsequently applied to the preprocessed content, it will be 100% effective in embedding the watermark.

3.2 Preprocessing for a normalized correlation system

The same technique can be applied to more complex watermarking systems, such as those that use normalized correlation as a detection metric (see, for example, [12]). Here, the detector computes the normalized correlation between a received Work, $\mathbf{c}$, and a reference pattern, $\mathbf{w_r}$,

eliminate the problem entirely. However, lattice codes are inherently fragile against valumetric scaling distortions, which limits their applicability. Compensating for this limitation is a subject of on-going research [14, 18]. It is not clear that the issue of valumetric scaling can be solved without re-introducing the problem of unwatermarkable content.

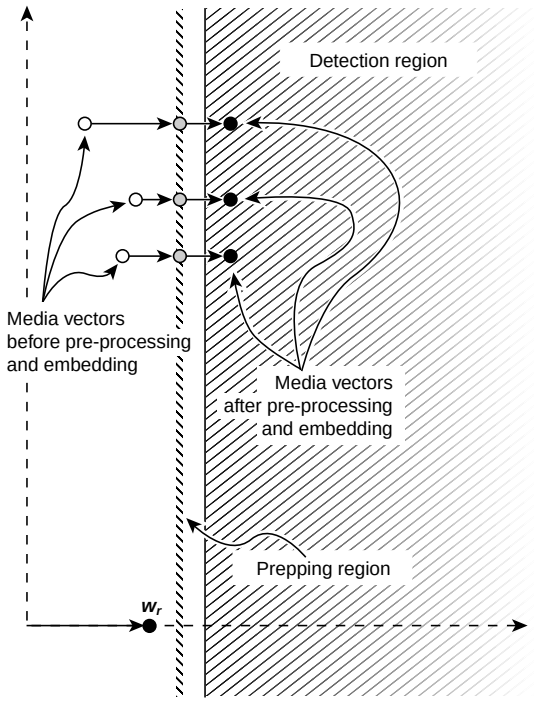


Figure 4: Geometry of the preprocessing and embedding.

as

$$z_{\mathrm{nc}} = \frac{\mathbf{c} \cdot \mathbf{w_r}}{\sqrt{(\mathbf{c} \cdot \mathbf{c})(\mathbf{w_r} \cdot \mathbf{w_r})}}. \tag{2}$$

This results in a conical detection region.

Here again, blind embedding can often successfully embed watermarks, but it fails in many cases. It is argued in [11, 20] that a more reliable method of embedding is to seek a fixed estimate of robustness. We can estimate robustness as the amount of white noise that may be added to the watermarked Work before it is likely to fall outside the detection region. This is given by

$$R^2 = \left(\frac{\mathbf{c} \cdot \mathbf{w_r}}{\tau_{\mathrm{nc}} \left| \mathbf{w_r} \right|} \right)^2 - \mathbf{c} \cdot \mathbf{c} \tag{3}$$

where τ_{nc} is the detection threshold that will be applied to the normalized correlation, and R^2 is the estimate of robustness (see [11] for a derivation of this equation). A fixed-robustness embedder that uses this estimate of robustness will employ a hyperbolic embedding region, as shown in Figure 5. Although such an embedder is preferable for many applications, it can be quite costly, as it not only requires examining the entire Work before embedding (which requires buffering), but also involves solving a quartic equation to find the closest point in the embedding region [9].

To obtain the reliability of a fixed-robustness embedder, while using a simple blind algorithm to embed, we can define a prepping region by shifting the embedding region outside the detection region. The distance that the embedding region must be shifted depends on the embedding strength that will be used by the blind embedder. This is shown in Figure 6. Here, the prepping region is a hyperboloid that lies entirely outside the detection cone. When a blind embedder is applied to a preprocessed Work (grey circle), the Work is moved into the detection region,

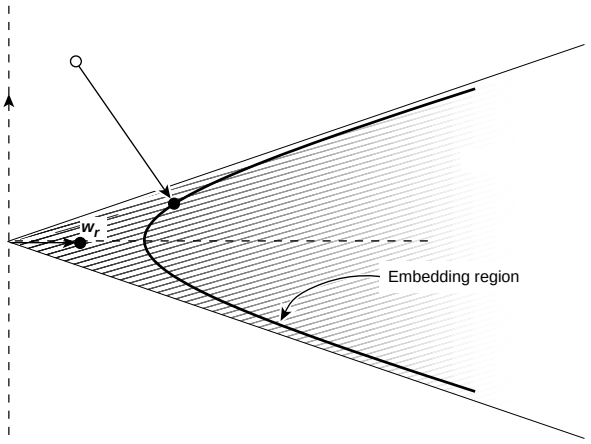


Figure 5: Behavior of a fixed-robustness embedder for a normalized-correlation based watermark. The shaded area is a conical detection region obtained by applying a threshold to the normalized correlation between a Work and a reference pattern. The embedding region, which comprises all the points that can survive a specified amount of white noise, is a hyperboloid within this cone.

so that the resulting watermarked Work (black circle) lies on the desired contour of constant robustness (dotted line).

Note that, if the embedding strength that will be used during blind embedding is too low, the shifted embedding region might overlap with the detection region. This would not be satisfactory as a prepping region, since it would lead to false positives. To solve this problem, we can simply remove a portion of the shifted embedding region from consideration during preprocessing. The preprocessor would move each Work to the closest point on the shifted hyperboloid that lies sufficiently far outside the detection region. This is illustrated in Figure 7.

3.3 Preprocessing for multiple bit watermarks

The two systems described above apply preprocessing to simple, zero-bit watermarks. That is, the detectors in these systems report whether the watermark is present or absent, but do

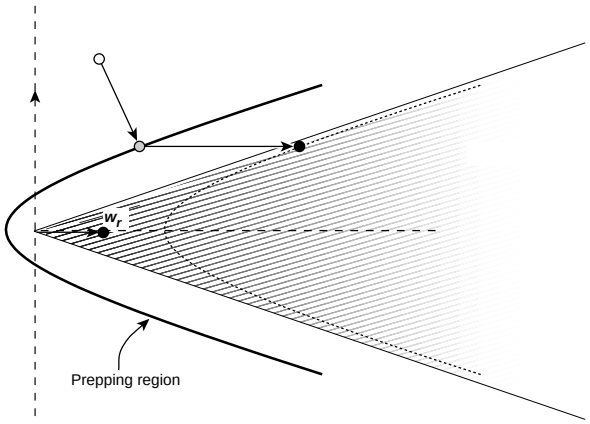


Figure 6: Preprocessing to obtain constant robustness when a blind embedder is applied. The open circle shows an unwatermarked Work. The grey circle shows the effect of preprocessing that Work. And the black circle shows the Work obtained by applying a blind embedder to the preprocessed Work.

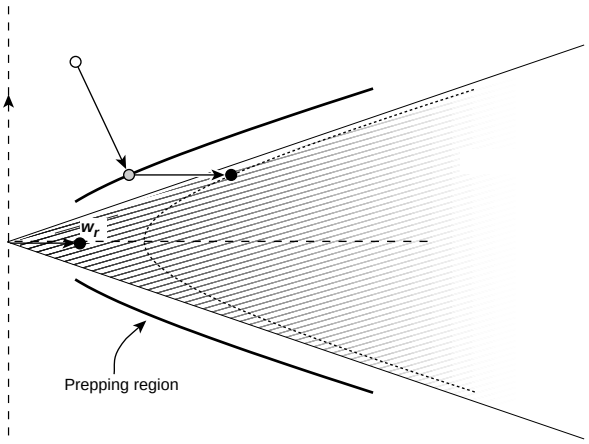


Figure 7: Preprocessing for constant robustness with a weak blind embedder. Because the embedding strength used during blind embedding will be small, the shifted hyperboloid does not lie entirely outside the detection region. This is solved by ignoring a portion of the shifted hyperboloid during preprocessing. The result is a prepping region that is only part of the hyperboloid.

not distinguish between different watermark messages, so the watermark carries zero-bits of payload information. If we have a system that can embed several different watermark patterns, representing different messages, we must modify our preprocessing method accordingly.

In the simplest case, we might have a system with two possible messages, or 1 bit of payload. For a message of $m = 1$, we might embed a reference mark, $\mathbf{w_r}$. For $m = 0$, we might embed the negation of the reference mark, $-\mathbf{w_r}$. The detector would check for presence of both the positive and negative watermark, reporting the corresponding message if one of them is found. Such a system, then, would define two disjoint detection regions, one for each message.

To ensure that blind embedding will succeed in embedding *any* of the possible messages, the preprocessor must move content to a prepping region that is the *intersection* of appropriate prepping regions for all the messages. For example, consider a 1 bit system using normalized correlation as its detection metric, as illustrated in Figure 8. The two detection regions in this case would be two opposing cones. A fixed-robustness embedder, when embedding $m = 1$, would move each Work to a hyperbolic embedding region within the positive cone. When embedding $m = 0$, it would move each Work to an embedding region within the negative cone. Shifting each of these embedding regions according to the effect of a blind embedder gives us two possible prepping regions – one that ensures the blind embedder can embed message $m = 1$, and one that ensures it can embed message $m = 0$. Only a Work in the intersection of these two regions will allow successful embedding of either message.

Note that the two points in the prepping region shown in Figure 8 actually correspond to a high-dimensional hypersphere in media space. This can be seen by realizing that, in three dimensions, the two points are rotated around the x-axis of the figure to obtain a 2-dimensional circle. In four dimensions, this circle is rotated to obtain a sphere, and in N-dimensional media

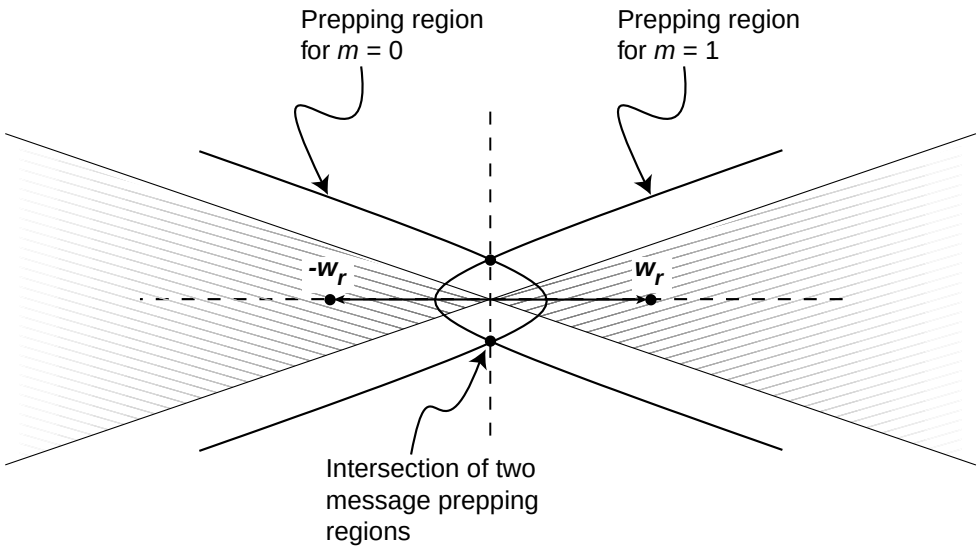


Figure 8: Preprocessing for a one-bit, normalized-correlation watermarking system. The one-bit detector defines two conical detection regions: one centered around $\mathbf{w_r}$, for $m = 1$, and one around $-\mathbf{w_r}$, for $m = 0$. For each detection region, there is a *message prepping* region of content in which a blind embedder can embed the corresponding message. The overall prepping region is the intersection of these two message prepping regions, which comprises two points in this 2-dimensional figure.

space, it is rotated into an $(N-1)$-dimensional sphere. Thus, although the figure appears to define a prepping region of only two points, the actual prepping region is a high-dimensional surface, and, with appropriate watermark extraction techniques, it is possible to implement a preprocessor that does not introduce too much distortion (see Section 5).

A problem that might arise is that the prepping regions for the separate messages do not intersect. This would occur if the embedding strength used by the blind embedder is too weak. In such a case, it would be impossible to perform the type of preprocessing we are proposing here. However, this is a pathalogical case regardless of whether preprocessing is employed, as it means there is no single Work into which the blind embedder can embed all possible messages. For every Work, there is at least one message that the blind embedder cannot embed. Thus, this would be a case of an unacceptable blind embedder that cannot be made acceptable by preprocessing.

It might appear, from Figure 8, that we will necessarily have the problem of non-overlapping message-prepping regions when we introduce even one additional message. After all, there is no way to place an additional cone in the figure so that it's message-prepping region intersects with either of the two points of intersection illustrated. But this is an illusion caused by our limited, 2-dimensional figure. To understand that many more than two detection cones can have intersecting message-prepping regions, imagine that the center lines of the cones (reference marks) all lie on a single plane in a 3-dimensional space. The message-prepping regions for all these cones can intersect at two points, one above the plane and one below it. As in the case of the two-point prepping region of Figure 8, these two points in 3-space correspond to a high-dimensional hypersphere in media space.

4 Performance considerations

The above discussion of preprocessing has focused on the robustness of the watermark embedded by a simple embedder. However, by introducing the preprocessing step, we have introduced some new questions regarding the fidelity, robustness, and security in the overall system. Can we obtain satisfactory fidelity in both the preprocessed and watermarked media? What happens if the preprocessed Work is distorted by normal processing before the watermark is embedded? Does preprocessing introduce any new security risks? Each of these questions is addressed in turn, below.

4.1 Fidelity

In watermarking systems that do not involve preprocessing, the embedder must create a watermarked Work that lies within some region of acceptable fidelity around the original. When we introduce a preprocessing step, we must now find *two* new Works within the region of acceptable fidelity: the preprocessed Work and the watermarked Work. These must be separated by the effect of the simple embedder.

In our experience, finding these two Works has not been difficult. This is not surprising, as the simple embedder will usually be designed to introduce very little fidelity degradation. Thus, the preprocessed and watermarked Works will be perceptually very similar, and if the fidelity of one is acceptable, the fidelity of the other is unlikely to be much worse.

Furthermore, the application may be designed in such a way that the preprocessed Work is never actually seen. For example, in the DiVX application, video never leaves the player without having a watermark embedded. In this case, the fidelity of the preprocessed video would be irrelevant, and we would only be concerned with the fidelity of watermarked video.

The problem of maintaining this fidelity is little different than that in a system that does not entail preprocessing.

4.2 Robustness

In some applications, the preprocessed Work might be expected to undergo some normal processing before the simple watermark embedder is applied. This would not be the case in the DiVX application, as the embedder is applied immediately after the video is read off the disk, but in the DVD application, it is expected that preprocessed video will be broadcast via television before it reaches the watermark embedder in a DVD recorder. Such broadcasting might entail lossy compression and analog distortions. This raises the question of whether these distortions will ruin the preprocessing, so that subsequent embedding fails.

In the case of additive distortion, where the distortion is independent of the Work being distorted, the performance of a system with a blind embedder is the same whether the distortion is applied before or after watermark embedding. If the distortion is applied first, it doesn't change the behavior of the embedder, and if the embedding is applied first, it doesn't change the nature of the noise. Thus, if the system is designed to yield a watermark that is robust to such noise, the use of preprocessing will not reduce its robustness.

However, many, if not most, distortions that can be expected are not independent of the Work. In these cases, there is a difference between applying the distortion before or after watermark embedding. For some distortions, this difference is small, and systems designed to be robust against additive noise will usually be reasonably robust against them. But other distortions are highly dependent on the Work to which they are applied, and these can represent a serious problem to a system employing preprocessing.

Perhapse the most severe example of such a class of distortions for video is the class of geometric distortions – translation, scaling, rotation, etc. If any of these distortions is applied to preprocessed video, it can desynchronise the preprocessing from the watermark embedding, causing the embedder to be no more effective than it would be on unpreprocessed video.

This is a problem that probably cannot be solved in a general way. In the DVD application, however, it can be solved by taking advantage of the detector for the `copy_once` mark. This detector must be robust against the same geometric distortions that might cause `copy_no_more` embedding to fail. Robustness against geometric distortions is usually attained by detecting those distortions and inverting them before watermark detection. Thus, a description of the distortions can be made available to the `copy_no_more` embedder. The embedder can then apply them to the watermark pattern, so that the pattern is once again synchronized with the preprocessing.

4.3 Security

The final question to be addressed is whether a system that depends on preprocessing is necessarily less secure than one that does not. This question is of particular interest in the two example applications of Section 2, as they are both intended to deter unauthorized copying.

The main, novel security risk that preprocessing might introduce is a risk that adversaries might modify preprocessed media so that subsequent embedding fails. This assumes, of course, that the adversary has access to the preprocessed media before the embedder is applied. It is possible to imagine applications of preprocessing in which the adversary has no such access. For example, we might build a streaming media server that puts a unique watermark into each stream. The stored media could be preprocessed to facilitate the use of inexpensive, real-time

embedders. As all the embedding occurs before the media reaches the customer, the adversary will not have access to anything unwatermarked.

Unfortunately, in the DVD and DiVX applications, the adversary must be assumed to have access to unwatermarked video. In the case of DiVX, this would require hacking the player to disable or bypass the embedder. In the DVD application, unwatermarked video is broadcast in the clear. In these cases, the adversary may very well be able to modify the video so that the embedder will fail.

The question, however, is why would the adversary bother? Presumably, his aim is to make a copy of the video that does not contain the watermark. If he has access to the unwatermarked video, he needn't modify it – he can just copy it. In the case of DVD, this would require a non-compliant or hacked recorder that would not embed a mark. In the case of DiVX, this could be done with any recorder, once the DiVX player has been hacked. Thus, if the unwatermarked video is available to the adversary, the risk introduced by reliance on preprocessing is arguably irrelevant.

A second risk in the types of systems being discussed here arises from the weakness of the embedder itself. Simple embedding algorithms are more likely to be easily hacked. This risk is particularly high if the adversary has access to an embedder, and can compare unwatermarked with watermarked media, which is the case in the DVD and DiVX applications. But this risk is not a consequence of reliance on preprocessing. The simplicity of the embedder, and its availability to the adversary, are dictated by the application. Preprocessing is merely a trick that makes such an application feasible.

Our conclusion, then, is that preprocessing may introduce some novel security risks, but these only arise in application settings where security is extremely weak anyway. However, it

must be noted that weak security can still be valuable. The proposed DVD system would add a level of deterrence to certain illegal copying which is presently entirely undeterred. If enough people are unwilling to bother breaking the system, the cost of that system may be justified.

5 An implementation

To illustrate the preprocessing technique, we implemented a preprocessor for the E_BLK_BLIND D_BLK_CC image watermarking system described in [9]. This is a one-bit, normalized-correlation system which operates in a linear projection of image space.

E_BLK_BLIND is a simple blind embedder. Although its description and implementation in [9] are a bit more complicated (to allow easy modifications into more sophisticated embedders), it essentially just adds or subtracts a scaled, tiled watermark pattern to the image. It takes as input an image, $\mathbf{c}$, to be watermarked, a message of either $m = 1$ or $m = 0$, an embedding strength, α, and an 8×8 reference mark, $\mathbf{w_r}$. If $m = 1$, the embedder adds $\alpha\mathbf{w_r}$ to each 8×8 block in the image. If $m = 0$, it subtracts $\alpha\mathbf{w_r}$ from each block.

The D_BLK_CC detection algorithm consists of two steps. In the first step, a mark vector, $\mathbf{v}$, is extracted from an image, $\mathbf{c}$, by averaging together 8×8 blocks to form one array of 64 values, as illustrated in Figure 9. The mark vector, $\mathbf{v}$, is given by

$$\mathbf{v}[i,j] = \frac{1}{B}\sum_{x=0}^{w/8}\sum_{y=0}^{y=h/8}\mathbf{c}[8x+i, 8y+j] \tag{4}$$

where $0 \le i < 8$ and $0 \le j < 8$ and w and h are the width and height of the image.

In the second step, the correlation coefficient[7], z_{cc}, is computed between the averaged 8×8

[7]The correlation coefficient between two vectors is just their normalized correlation after projection into a space with one fewer dimension (see [9]). Thus, the detector computes the normalized correlation in a 63-dimensional space.

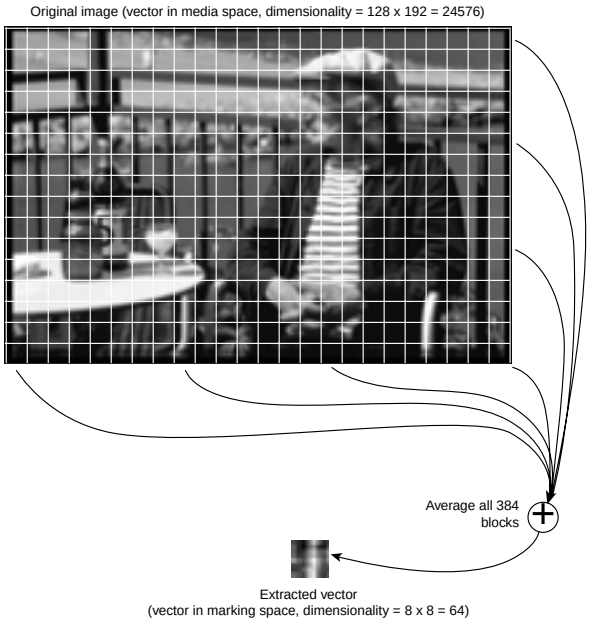


Figure 9: Watermark extraction procedure.

block, $\mathbf{v}$, and the reference mark, $\mathbf{w_r}$. That is,

$$z_{\text{cc}} = \frac{\tilde{\mathbf{v}} \cdot \tilde{\mathbf{w}}_\mathbf{r}}{\sqrt{(\tilde{\mathbf{v}} \cdot \tilde{\mathbf{v}})(\tilde{\mathbf{w}}_\mathbf{r} \cdot \tilde{\mathbf{w}}_\mathbf{r})}} \tag{5}$$

where $\tilde{\mathbf{v}} = (\mathbf{v} - \bar{\mathbf{v}})$, $\tilde{\mathbf{w}}_\mathbf{r} = (\mathbf{w_r} - \bar{\mathbf{w}}_\mathbf{r})$ and $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}_\mathbf{r}$ are the means of $\mathbf{v}$ and $\mathbf{w_r}$. It compares z_{cc} against a detection threshold, τ_{cc}. If $z_{\text{cc}} > \tau_{\text{cc}}$, it reports that message $m = 1$ has been embedded. If $z_{\text{cc}} < -\tau_{\text{cc}}$, it reports that message $m = 0$ has been embedded. Otherwise, it reports that there is no watermark present.

We implemented a preprocessor for this system according to the principles described in Section 3.3 and illustrated in Figure 8. The preprocessor performs the following steps

1. Extract a mark vector, $\mathbf{v_o}$ from the unwatermarked Work, in the same manner as the detector.

2. Identify a 2-dimensional plane that contains $\mathbf{v_o}$ and the reference mark, $\mathbf{w_r}$. The plane is described by two, orthogonal, unit vectors $\mathbf{X}$ and $\mathbf{Y}$, obtained by Gram-Schmidt orthonormalization [16]:

$$\mathbf{X} = \frac{\mathbf{w_r}}{\sqrt{\mathbf{w_r} \cdot \mathbf{w_r}}} \tag{6}$$

$$\mathbf{Y_0} = \mathbf{v_o} - (\mathbf{v_o} \cdot \mathbf{X})\mathbf{X} \tag{7}$$

$$\mathbf{Y} = \frac{\mathbf{Y_0}}{\sqrt{\mathbf{Y_0} \cdot \mathbf{Y_0}}}. \tag{8}$$

(Note, $\mathbf{Y_0}$ here is a temporary vector.)

3. Project $\mathbf{v_o}$ into the $\mathbf{X}, \mathbf{Y}$ plane:

$$x_{\mathbf{v_o}} = \mathbf{v_o} \cdot \mathbf{X} \tag{9}$$

$$y_{\mathbf{v_o}} = \mathbf{v_o} \cdot \mathbf{Y} \tag{10}$$

4. Find the point in the prepping region, $\left\langle x_{\mathbf{v_p}}, y_{\mathbf{v_p}} \right\rangle$, that is closest to $\langle x_{\mathbf{v_o}}, y_{\mathbf{v_o}} \rangle$. As shown in Figure 8, the prepping region in this 2-dimensional plane comprises only two points. Since $y_{\mathbf{v_o}}$ is guaranteed to be positive, the upper of these two points will always be the closest to $\langle x_{\mathbf{v_o}}, y_{\mathbf{v_o}} \rangle$. Thus, $x_{\mathbf{v_p}} = 0$, and $y_{\mathbf{v_p}}$ is a positive value chosen to ensure that blind embedding will yield the desired level of robustness. To find $y_{\mathbf{v_p}}$, first note that, in the $\mathbf{X}, \mathbf{Y}$ plane, the watermark vector, $\mathbf{w_r}$, will be either $\langle k, 0 \rangle$ or $\langle -k, 0 \rangle$, depending on whether we wish to embed a 1 or a 0. Here, k is the magnitude of the watermark reference pattern, which was $\sqrt{N}$ in our experiments, where N is the size of the watermark reference pattern, i.e. $N = 64$. After the blind embedder is applied with a strength of α, we will obtain $\mathbf{v_w} = \mathbf{v_p} + \alpha \mathbf{w_r}$, which gives us, in the $\mathbf{X}, \mathbf{Y}$ plane, either $\left\langle \alpha k, y_{\mathbf{v_p}} \right\rangle$ or $\mathbf{v_w} = \left\langle -\alpha k, y_{\mathbf{v_p}} \right\rangle$.

By letting $\mathbf{w_r} = \langle \pm k, 0 \rangle$, $\mathbf{c} = \left\langle \pm \alpha k, y_{\mathbf{v_p}} \right\rangle$, and $\tau_{\text{nc}} = \tau_{\text{cc}}$ in Equation 3, and solving for $y_{\mathbf{v_p}}$, we obtain

$$y_{\mathbf{v_p}} = \sqrt{\frac{\alpha^2 k^2 (1 - \tau_{\text{cc}}^2)}{\tau_{\text{cc}}^2} - R^2} \tag{11}$$

where R^2 is the desired robustness.

5. Obtain a preprocessed mark vector, $\mathbf{v_p}$, by projecting $\left\langle x_{\mathbf{v_p}}, y_{\mathbf{v_p}} \right\rangle$ back into 64-dimensional space:

$$\mathbf{v_p} = x_{\mathbf{v_p}} \mathbf{X} + y_{\mathbf{v_p}} \mathbf{Y} \tag{12}$$

6. Invert the original extraction operation on $\mathbf{v_p}$ to obtain the preprocessed cover Work, $\mathbf{c_p}$. This is done by simply adding $\mathbf{v_p} - \mathbf{v_o}$ to each block of the image.

To test these procedures, we first tested the watermarking system on original images that had not been preprocessed, using a weak embedding strength of $\alpha = 0.5$. Watermarks of $m = 1$ and $m = 0$ were embedded in each of 2000 images from the Corel image database [7]. Each image was 256 by 384 pixels, and $k = 8$. Figure 10 shows the resulting detection values. The dotted line is a histogram of detection values for unwatermarked images, and each of the solid lines shows detection values for one of the embedded messages. With a detection threshold of $\tau_{\text{cc}} = 0.55$, this succeeded in embedding watermarks in just over 45% of the trials.

Next, we applied the preprocessor to each of the 2000 images, with $\tau_{\text{cc}} = 0.55$, $\alpha = 0.5$ and $R = 30$, and ran the same test again. The results are shown in Figure 11. As expected, application of the blind embedder to preprocessed images succeeded in embedding watermarks in 100% of the trials. In addition, the detection values obtained from preprocessed images before embedding a watermark are very narrowly distributed around 0. This indicates that they are less likely to yield false positives than are unpreprocessed images. In some applications, if we

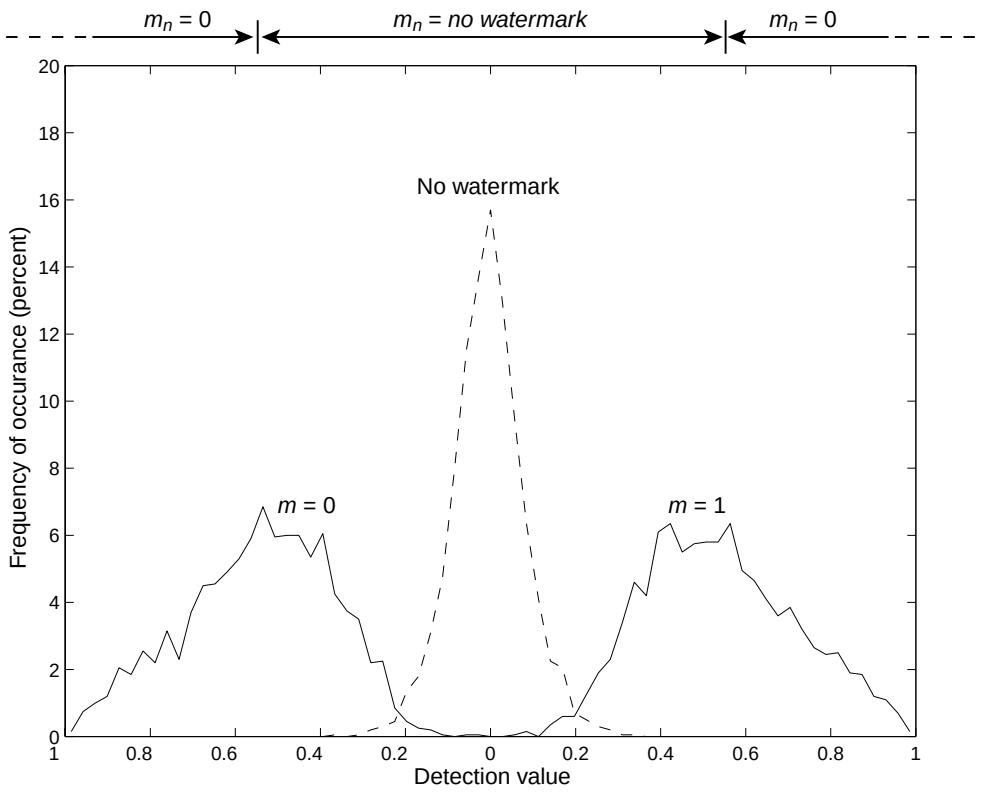


Figure 10: Results of the watermarking system with no preprocessing and $\alpha = 0.5$.

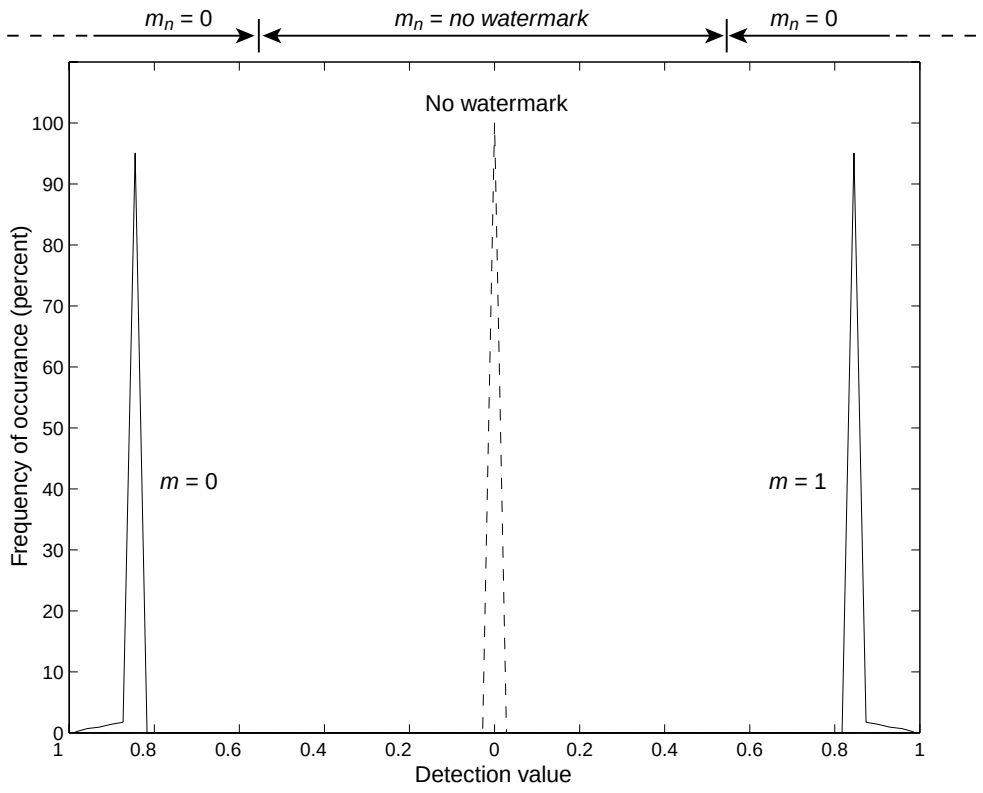


Figure 11: Results of the watermarking system applied to preprocessed images.

can guarantee that the detector will never be run on unpreprocessed images, we could take advantage of this to lower the detection threshold, thereby obtaining even better robustness.

The question arises whether we could obtain equally good results, with the same fidelity, by just increasing the embedding strength used during blind embedding. Blind embedding alone, with no preprocessing, yields an average mean-squared-error between marked and unmarked images of exactly α (because of the way we scaled $\mathbf{w_r}$). Preprocessing, however, introduces additional fidelity degradation. The average mean-squared-error between original images and images that have been both preprocessed and watermarked was just under 1.04. If, instead of applying preprocessing, we simply increased α to 1.04, we would obtain the same fidelity impact

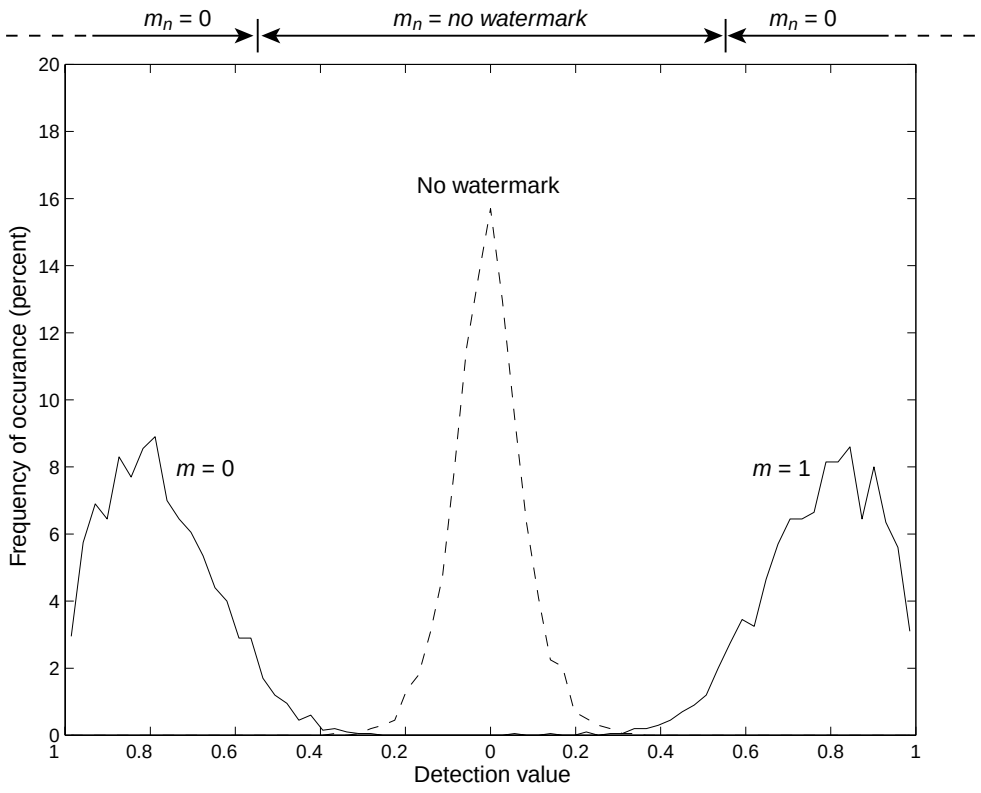


Figure 12: Results of the watermarking system with no preprocessing and $\alpha = 1.04$.

as preprocessing plus embedding, but we would have substantially stronger watermarks than with $\alpha = 0.5$. Would this yield 100% effectiveness without preprocessing?

Figure 12 shows the results of applying the blind embedder to unpreprocessed images with $\alpha = 1.04$. Although this performance is vastly better than that of Figure 10, it is still inferior to the performance obtained with preprocessing. With this higher value of α, blind embedding still failed to embed watermarks in just under 6% of the trials.

Of course, since we can assume that we have substantial computing power available during preprocessing, we can improve on the fidelity impact of preprocessing by applying more sophisticated algorithms, such as perceptual modeling. Such improvements would increase the disparity

between watermarking with and without preprocessing.

6 Conclusion

There are several watermarking applications in which a potentially very large number of embedders must be deployed under severe computational constraints that limit performance. In order to attain the performance of sophisticated embedding algorithms, and yet maintain the simple, inexpensive embedder, we propose preprocessing media before it is released. Most of the computational cost is shifted to the preprocesing stage where it is assumed that significant resources are available.

Our proposal is applicable in settings where content can be modified before it reaches the watermark embedders. Two examples of such applications are the transaction-tracking system deployed by the DiVX corporation, and the proposed Galaxy watermarking system for copy protection of DVD video.

Before preprocessing, unwatermarked Works can be geometrically thought of as being randomly distributed in a high dimensional vector space. Within this space lies a detection region – Works falling within this region are said to be watermarked. Unwatermarked Works are seldom if ever found in the detection region. Traditional embedding algorithms seek to add a watermark pattern to a Work in order to move the Work into the detection subspace, subject to fidelity and robustness constraints. During the preprocessing stage suggested here, a signal is added to a Work such that the preprocessed Work lies on a predetermined surface near, but outside of the detection region. That is, the unwatermarked, but preprocessed Works are no longer *randomly* distributed in the high dimensional space but lie in a well-defined region.

This preprocessing step provides two main advantages. First, since preprocessed Works lie on

a well-defined surface, near yet outside of the detection region, simple embedding techniques are sufficient to watermark the Works with good fidelity and robustness. Second, the computational cost associated with the preprocessing step is not borne by the embedders. Instead, content creators bare this cost, the preprocessing being performed by dedicated devices located with content creators. Thus, the performance of the overall system need no longer be constrained by the computational budget allocated to the embedder.

A third possible advantage of preprocessing is that it can reduce the probability of false positives. This results from the preprocessor ensuring that all Works are at least a certain distance outside the detection region. However, this advantage can only be exploited in applications where the watermark detector will never be applied to unpreprocessed content.

We have implemented a preprocessor for a simple, 1-bit watermarking system with blind embedding. Tests on 2000 images show that preprocessing significantly improves performance of the embedder.

References

[1] Sony Corporation of America *et al* and Universal City Studios, Inc *et al*. United States Court of Appeals for the Ninth Circuit, 1984.

[2] J. A. Bloom, I. J. Cox, T. Kalker, J-P Linnartz, M. L. Miller, and B. Traw. Copy protection for DVD video. *Proc. IEEE*, 87(7):1267–1276, 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.

[4] B. Chen and G. W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. In *Proc. Int. Symp. Inform. Theory (ISIT-2000)*, 2000.

[5] 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.

[6] Jim Chou and Kannan Ramchandran. Robust turbo-based data hiding for image and video sources. In *IEEE Int. Conference on Image Processing*, 2002.

[7] Corel Stock Photo Library 3, Corel Corporation, Ontario, Canada.

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

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

[10] I. J. Cox, M. L. Miller, and J. A. Bloom. Method for data preparation and watermark insertion. *United States Patent*, 6,332,194, 2001.

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

[12] I.J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for images, audio and video. In *IEEE Int. Conference on Image Processing*, volume 3, pages 243–246, 1996.

[13] J. F. Delaigle, C. De Vleeschouwer, and B. Macq. Watermarking algorithm based on a human visual model. *Signal Processing*, 66(3):319–335, 1998.

[14] J. J. Eggers, R. Bäuml, and B. Girod. Estimation of amplitude modifications before scs watermark detection. In *Proc. of SPIE on Security and Watermarking of Multimedia Contents*, volume 4675, 2002.

[15] 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.

[16] A.M. Erisman I. S. Duff and J.K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, 1986.

[17] T. Kalker. System issues in digital image and video watermarking for copy protection. In *IEEE Int. Conf. on Multimedia Computing and Systems*, volume 1, pages 562–567, 1999.

[18] K. Lee, K. A. Moon, D. S. Kim, and T. Kim. Em estimation of scale factor for quantization-based audio watermarking. In *2nd International Workshop on Digital Watermarking*, 2003.

[19] M. L. Miller. Watermarking with dirty-paper codes. In *IEEE International Conference on Image Processing*, September 2001.

[20] 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.

[21] M. L. Miller, G. Doërr, and I. J. Cox. Dirty-paper trellis codes for watermarking. In *IEEE Int. Conference on Image Processing*, 2002.

[22] Raymond B. Wolfgang, C. I. Podilchuk, and E. J. Delp. Perceptual watermarks for digital images and video. *Proc. of the IEEE*, 87(7):1108–1126, 1999.