Robust Image Watermarking with the Randlet Transform

Michael Malkin
mikeym@stanford.edu
Gates Building 4B
Stanford, CA, 94305

Ramarathnam Venkatesan
venkie@microsoft.com
One Microsoft Way
Redmond, WA, 98052

Abstract

The randlet transform is a new transform based on randomly-chosen basis functions. The randomized nature of the transform lends itself to watermarking applications, especially in the difficulty of an attacker discovering the watermark and in robustness against perceptually insignificant image modifications, including malicious attacks. Watermark embedding is done by quantizing the values of randlet transform coefficients. Since the basis functions are nonorthogonal, an optimization routine is used to minimize the distortion induced in order to quantize the coefficients. The randomization inherent in the randlet transform means that image modifications appear as noise in the transform coefficients, lending to good robustness and security characteristics, especially when used with soft decoding.

1 Introduction

Image watermarking involves hiding data in a preexisting image. For our purposes, the watermark should be robust to small changes in the image while causing minimal visible changes in the image. The randlet transform is a new transform with randomly-chosen basis functions which is well suited to the watermarking problem. The basis functions are robust to attacks on the watermark and at the same time are minimally perceptible when embedded in a watermark. Furthermore, an adversary who does not know the random basis functions is at a great disadvantage when attacking the watermark. We will examine the properties of the randlet transform as applied to image watermarking, using a linear optimization algorithm to modify the randlet transform coefficients while inducing a minimum of noise in the image. At the watermark decoder we consider both hard and soft decoding.

We consider two types of watermarking. With verification watermarking, the problem is to recognize if a given image has been watermarked with a given randlet basis. In this way, for example, ownership of an image can be verified by proving that a specific, secret randlet basis has been watermarked into the image. The second type of watermarking we consider is data watermarking, where multiple bits of data are embedded into an image and are extracted by a watermark detector. This type of watermarking can also be used to prove ownership, but it can also be used to embed sideband information into an image.

The security of a randlet watermark is based on a secret key which is used to pseudorandomly generate the randlet basis and randomized quantizers that are used. An attacker knows the randlet basis has a much better chance of erasing the watermark. An attacker with knowledge of the randlet basis and the random quantizers can forge their own watermarks.

For related work, Malkin and Venkatesan [6] present the randlet transform in detail and describe applications to image hashing and image identification. The full randlet transform is similar to the matching pursuits algorithm [7]. Mihcak et al. [5] use a similar watermarking scheme, but with less structure used in defining the watermark. Our work is also similar to quantization index modulation as developed by Chen and Wornell [2], but we use non-orthogonal basis functions, which introduces difficulties when quantizing transform coefficients. Fridrich [3] also considers the use of randomness in image watermarking.

Section 2 introduces the randlet transform. Section 3 contains a full description of our watermarking system. Finally, Section 4 describes real-world tests of the system.

2 The Randlet Transform

What follows is a short description of the randlet transform, focusing on generating a randlet basis and using a randlet basis to embed and detect watermarks. For a more complete explanation of the randlet transform, see [6].

In the randlet transform, basis functions called *randlets* are generated pseudo-randomly from a secret key. The transform itself is an iterative process. Each randlet is projected onto an image to find the corresponding transform coefficient, then the projection is subtracted from the image. This process is known as the *full* randlet transform. In some situations it is not necessary to subtract the projection of each randlet from the image, in which case the transform is known as the *forward* randlet transform. The forward randlet transform is used in randlet-based image-recognition systems such as [6] and in the watermarking work in this paper.

During a full randlet transform, a search can be performed by perturbing randlets to find the (nearby) projection with the largest power. These perturbations can include small translations and rotations, increases and decreases in frequency, etc. Such perturbations introduce side-information that increases the size of the transform data, but can dramatically speed the transform's convergence.

2.1 Generating a Randlet Basis

Any two-dimensional function with compact or effectively-compact support can be a randlet, and any collection of such functions can be considered a randlet basis. In practice, however, the addition of some amount of structure greatly increases the speed and lowers the memory cost of the transform and inverse transform, while experimentally imposing no cost. All of the randomness used to form a randlet basis is generated pseudo-randomly with the secret key.

For image watermarking applications, randlets are discrete, two-dimensional functions. First, a set of *mother randlets* is chosen. These are the templates from which other randlet are chosen. The mother randlets are then randomly chosen, then randomly scaled, rotated, and otherwise modified to form *base randlets*, which are then normalized. The base randlets are randomly translated to form the randlets, sometimes called *final randlets*, which may overlap. All of the random decisions (choice of mother randlets, rotations, translations, etc.) are made pseudo-randomly using the secret key according to user-defined distributions. Note that a randlet basis will not necessarily span the set of all images. Below are some example randlets.

Gaussian Randlet: Translations, rotations, and scalings of: $e^{\alpha x^2+\beta y^2}$.

Mexican Hat Randlet: Translations, rotations, and scalings of: $y^2 e^{\alpha x^2 + \beta y^2}$.

Random Randlet: Translations of elliptical regions where each pixel value is chosen from a normal distribution with standard deviation 1, and the result is normalized.

Smooth Random Randlet: Random randlets that have been low-pass filtered.

Curvelet Randlet: Curvelets [1] may be used as randlets.

Wavelet Randlet: Translations, rotations, and scalings of: $w(y)e^{\alpha x^2}$,where $w(y)$ is a wavelet.

The most important randlets in applications involving images are *perceptual* randlets, which are randlets that have few high-frequency components. In watermarking applications, these randlets result in less perceptual distortion, and are more robust to desynchronization attacks, especially geometric attacks like rotation and cropping, because the they are smoothly varying.

A randlet basis is generated for a specific image size. A set of modifications for each mother randlet is chosen randomly, effectively deciding the base randlets. The final randlets are then chosen by selecting the among base randlets according to some distribution, and translating the result uniformly across the image size. Randlets that do not have compact support must still have effectively-compact support, meaning that they approach zero quickly enough that they can be truncated to speed up transform and inverse transform operations and reduce memory requirements.

2.2 Performing a Randlet Transform

Let $\mathbf{A}$ be an image, $\mathbf{r}_1, ..., \mathbf{r}_n$ be randlets, and $c_1, ..., c_n$ be randlet transform coefficients. To find the randlet transform of $\mathbf{A}$, iteratively consider each randlet. In a *full* randlet transform, each randlet is projected onto the image and the projection is subtracted before the next randlet is projected. Let $\mathbf{R}_j$ be the *residual* after the projection of randlet j is subtracted, and define $\mathbf{R}_0 = \mathbf{A}$. Then for a full randlet transform,

$$c_j = \mathbf{R}_{j-1} \cdot \mathbf{r}_j, \qquad \mathbf{R}_j = \mathbf{R}_{j-1} - c_j \mathbf{r}_j, \qquad j \in \{1, ..., n\}.$$

The variant *forward* randlet transform (used in watermarking) is more simple than the full randlet transform:

$$c_j = \mathbf{A} \cdot \mathbf{r}_j, \qquad j \in \{1, ..., n\}.$$

The inverse randlet transform also very simple and involves multiplying each randlet by its corresponding coefficient and adding them together:

$$\mathbf{A} = \sum_{j=1}^{k} c_j \mathbf{r_j}.$$

3 Randlet Watermarking

The basis of our watermarking algorithm is the quantization of transform coefficients, similar to ideas presented by Mihcak et al. [5] and Chen and Wornell [2]. We choose quantization watermark embedding over spread spectrum watermark embedding so that the image itself does not act as interference with the watermark.

To watermark an image, the randlet transform of the image is found using a secret randlet basis. The transform coefficients are quantized using secret quantizers and the image is modified so that the transforming the watermarked image will yield the quantized coefficients. Because the randlet basis functions are nonorthogonal, there will be

conflicting goals in this process. For example, changing the image so that one coefficient is quantized may affect the value of another coefficient. We use a linear optimization algorithm is to achieve the quantization while reducing the embedding distortion. Watermark detection involves taking the randlet transform and looking at how close each coefficient is to a quantization point on the secret quantizers.

We consider two types of watermarking. First is a verification watermarking, used only to tell if an image is watermarked or not. Second is a data watermarking, which can contain multiple bits of information. The two types of watermarks differ mainly in how they quantize the coefficients, although detection is also slightly different in each case.

For verification watermarks, a randomly-shifted uniform quantizer is chosen for each transform coefficient. To embed a watermark, a randlet transform is taken of an image and each transform coefficient is quantized to the nearest quantization point on its corresponding quantizer. Detection is done by taking the randlet transform of an image and examining how close each coefficient is to a quantization point on its corresponding quantizer. If enough coefficients are deemed to have been quantized, the image is considered watermarked.

For data watermarks, each randlet embeds a single bit. A randomly-shifted pair of uniform quantizers with alternating quantization points is chosen. One quantizer corresponds to a bit of 0 and the other to a bit of 1. To embed a watermark, a randlet transform is taken of an image and each coefficient is quantized to the nearest quantization point on one of the quantizers, depending on whether a 0 or a 1 is to be embedded. To detect a watermark, the randlet transform is taken of an image, and a bit is assigned a value based on which quantizer has the closest quantization point.

The randlet basis that is used for watermarking depends only on the secret key, not on the specific image being watermarked. For an image-dependent watermark, an image hash can be used to supplement the secret key. See Malkin and Venkatesan [6] for an illustration of how to do this robustly with randlets.

3.1 Definitions

We will consider watermarking an $n \times m$ pixel image with k randlets. The image is stored internally as an $N \times 1$ column vector, where $N = nm$. This vector is formed by stacking the columns of the image on top of each other. Let A be an image, and let the pixel at row y and column x be referenced as $A[x, y]$, $x \in [1, ..., n]$, $y \in [1, ..., m]$. Then the column-vector representation of A is:

$$\mathbf{A} = (A[1,1], ..., A[1,m], A[2,1], ..., A[2,m], A[n,1], ..., A[n,m])^T.$$

The randlets and watermark are also $n \times m$ pixel images stored as $N \times 1$ column vectors. For a randlet transform with k randlets, let $\mathbf{r}_1, ..., \mathbf{r}_k$ be randlets. The randlet transform produces a column vector of k coefficients, $\mathbf{c} = (c_1, ..., c_k)^T$ such that $c_j = \mathbf{r}_j^T \mathbf{A}$. Therefore, we can write the randlet transform T as a matrix of row vectors of randlets, $T = [\mathbf{r}_1, ..., \mathbf{r}_k]^T$. T is a $k \times N$ matrix.

3.2 Choosing Quantization Values

The first step in the randlet watermarking algorithm is to choose the the watermark. A randlet watermark is exists in the randlet transform coefficients. If these coefficients are close to quantization points on the secret watermarking quantizers then they are considered watermarked. An image is watermarked by modifying it in such a way that

the randlet transform coefficients are close to quantization points. We consider the case of quantizing each coefficient separately, but note that it is also possible to perform the quantization with vector quantizers.

Recall that we consider two types of watermarking, verification watermarking and data watermarking. For verification watermarking, a randomly-shifted uniform quantizer is chosen for each coefficient. Define Q_j to be the quantizer for coefficient j. Let Δ_j be the quantization step size and $\alpha_j \in [-\frac{\Delta}{2}, \frac{\Delta}{2}]$ be a random shift. Then the quantization points of Q_j are

$$Q_j = \{..., -2\Delta_j + \alpha_j, -\Delta_j + \alpha_j, \alpha_j, \Delta_j + \alpha_j, 2\Delta_j + \alpha_j, ...\}.$$

Let $Q(\mathbf{c})$ be the operation of quantizing every coefficient in $\mathbf{c}$ with the corresponding quantizer. For verification watermarking, then, the quantization operation is $Q(\mathbf{c}) = (Q_1(c_1), ..., Q_k(c_k))^T$.

For data watermarking, the quantizer above is split into two quantizers, Q_j^0 and Q_j^1, corresponding to embedding a 0 bit and a 1 bit, respectively. This is done by alternately assigning quantization points on the verification watermarking quantizer to Q_j^0 and Q_j^1. The result is that both are uniform quantizers with alternating quantization points, and each quantization point on one is midway between quantization points on the other. Furthermore, the pair of quantizers is randomly shifted. As with verification watermarking, for randlet j, let Δ_j be the quantization step size and $\alpha_j \in [-\Delta, \Delta]$ be a random shift. Then quantization points for Q_j^0 and Q_j^1 are

$$Q_j^0 = \{..., -4\Delta + \alpha_j, -2\Delta + \alpha_j, \alpha_j, 2\Delta + \alpha_j, 4\Delta + \alpha_j, ...\},$$

$$Q_j^1 = \{..., -3\Delta + \alpha_j, -\Delta + \alpha_j, \Delta + \alpha_j, 3\Delta + \alpha_j, ...\}.$$

Let $\mathbf{v} \in \{0,1\}^k$ be the secret data to be embedded and let v_j be bit j of $\mathbf{v}$. Given quantizers Q_j^0 and Q_j^1 for all $j \in \{1, ..., k\}$, define $Q(\mathbf{c})$ to be the operation of applying $Q_j^{v_j}$ to c_j, so the quantization operation is $Q_\mathbf{v}(\mathbf{c}) = (Q_1^{v_1}(c_1), ..., Q_k^{v_k}(c_k))^T$.

The type of quantization, either verification or data watermarking, should be obvious from the context.

3.3 Watermark Embedding

Because the forward randlet transform is linear, it is possible to represent it in matrix form. In practice such a representation is inefficient in both time and space, prohibitively so with large images or many randlets. However, the watermarking algorithm is conceptually related to the matrix formulation, so we will begin by considering the watermarking algorithm in matrix form, and then relate that to a more practical algorithm.

Image A has randlet transform $\mathbf{c} = \mathbf{TA}$. We choose the watermark to be the smallest $\mathbf{W}$ such that $\mathbf{T}(\mathbf{A} + \mathbf{W}) = Q(\mathbf{c})$. In other words, we would like $\mathbf{W}$ to be the min-norm solution. Because $\mathbf{T}(\mathbf{A} + \mathbf{W}) = \mathbf{c} + \mathbf{TW} = Q(\mathbf{c})$, we can write

$$\mathbf{TW} = (Q(\mathbf{c}) - \mathbf{c}).$$

Lemma 3.1 *The solution to minimize* $\mathbf{W}$ *such that* $\mathbf{T}(\mathbf{A} + \mathbf{W}) = Q(\mathbf{c})$ *is given by*

$$\mathbf{W} = \mathbf{T}^T(\mathbf{TT}^T)^{-1}(Q(\mathbf{c}) - \mathbf{c}). \tag{1}$$

As the min-norm solution in lemma 3.1 implies, the watermarked image, $\hat{\mathbf{A}}$, is

$$\hat{\mathbf{A}} = \mathbf{A} + \mathbf{W} = \mathbf{A} + \mathbf{T}^T(\mathbf{T}\mathbf{T}^T)^{-1}(Q(\mathbf{c}) - \mathbf{c}) \tag{2}$$

This is a simple solution, but since each randlet is the size of an image, $\mathbf{T}$ is a very large matrix. For example, to embed 500 coefficients into an image of size 512×512, $\mathbf{T}$ would be 500×262144 and have over 131 million elements. Since randlets have compact support, a large fraction of theses elements are zero. Furthermore, the non-zero entries are not scattered randomly around the matrix, but are highly ordered. We should, therefore, be able to improve the algorithm. We will do so by examining the nature of the min-norm solution.

Multiplication of an image by $\mathbf{T}$ is simply the randlet transform, $\mathbf{TA} = \mathbf{c}$, where each transform coefficient is the inner product of a randlet (one row of $\mathbf{T}$) with the image vector. Now consider multiplying an arbitrary vector $\mathbf{c}$ with the transpose of the transform matrix. $\mathbf{B} = \mathbf{T}^T\mathbf{c}$ is an image which is formed as a linear combination of randlets, with randlet j being weighted by c_j. In other words, this is the inverse randlet transform.

Now let us consider the watermark, $\mathbf{W} = \mathbf{T}^T(\mathbf{T}\mathbf{T}^T)^{-1}(Q(\mathbf{c}) - \mathbf{c})$. The first step in computing the watermark is to find $\mathbf{S} = \mathbf{T}\mathbf{T}^T$. If the element of $\mathbf{S}$ at row i, column j is denoted s_{ij}, then $s_{ij} = \mathbf{r}_i^T\mathbf{r}_j$. That is, the element at row i and column j is the inner product of randlet i and randlet j.

Examining $\mathbf{S}$ can greatly speed up the watermarking process. Since the randlets are normalized, $s_{ii} = 1$, and since the inner product is commutative, $s_{ij} = s_{ji}$. We therefore have to compute fewer than half of the elements in the matrix. Of the matrix elements that must be computed, $s_{ij} = 0$ if randlet i and randlet j do not overlap. Furthermore, the inner product of $\mathbf{r}_i$ and $\mathbf{r}_j$ only needs to be computed where randlet i and randlet j overlap, instead of computing the inner product of two length-N vectors, and N is very large.

Multiplying the $k \times 1$ vector $(Q(\mathbf{c}) - \mathbf{c})$ by the $k \times k$ matrix $\mathbf{S}^{-1} = (\mathbf{T}\mathbf{T}^T)^{-1}$ gives an $M \times 1$ vector of weights, which we will call $\mathbf{v}$:

$$\mathbf{v} = (\mathbf{T}\mathbf{T}^T)^{-1}(Q(\mathbf{c}) - \mathbf{c})$$

As was shown earlier, $\mathbf{T}^T\mathbf{v}$ is simply the inverse randlet transform of $\mathbf{v}$. That is, if $\mathbf{v} = (v_1, w_2, ..., w_k)^T$, then the result is an image formed by the sum of randlet j multiplied by w_j.

$$\mathbf{T}^T\mathbf{v} = \mathbf{T}^T(\mathbf{T}\mathbf{T}^T)^{-1}(Q(\mathbf{c}) - \mathbf{c}) = \sum_{i=1}^{k} v_j\mathbf{r_j}$$

We can find the inverse transform by multiplying each randlet by its corresponding weight and adding the results together. Since each randlet has compact support, and conversely because $\mathbf{T}^T$ is so large, this is generally much faster than performing the matrix multiplication.

So to find the optimum watermark, we first calculate $\mathbf{S} = \mathbf{T}\mathbf{T}^T$ by finding the inner product of every randlet with every other randlet. We form a vector of weights by inverting $\mathbf{S}$ and multiplying the desired change in transform coefficients by that inverse. Finally, we perform a randlet-by-randlet inverse randlet transform on vector of weights to produce the watermark, which is added to the original image.

Note that $\mathbf{T}\mathbf{T}^T$ must have a small condition number in order for the watermark embedding to be successful. If the condition number is too large there will be too much

embedding distortion. Generally, the condition number becomes large when too many very large randlets are used in the randlet basis, so using fewer or smaller randlets will prevent large condition numbers. Experimentally, this has not been problematic. Also note that it is possible to use linear regularization conditions with the linear optimization, or to use non-linear optimizations. However, we have found that when the condition number is small these steps are unnecessary.

Finally, it is possible to use a system similar to the distortion compensation mentioned by Chen and Wornell [2]. The quantization stepsize is increased, but the watermark is only partially embedded in the image. If δ is the quantization stepsize, and $\alpha < 1$ is the distortion compensation parameter, then define $\delta' = \delta/\alpha$ to be the new stepsize and $\mathbf{W}' = \alpha\mathbf{W}$ to be the new watermark. Distortion compensation generally improves performance.

3.4 Watermark Detection

We consider the case of blind watermark detection. The detector knows the randlet basis, the quantizers used, and (if detecting a data watermark) the secret data bit vector. The detector is given an image, possibly watermarked, possibly attacked, to perform detection upon. To detect a watermark, the randlet transform is taken of an image, and the resulting coefficients are checked to see how close they are to quantization points.

3.4.1 Verification Watermark, Hard Detection

For hard detection of a verification watermark, a coefficient is considered *marked* if it is within a given radius of any quantization point. If enough coefficients are marked, then the image is verified as being watermarked. Otherwise, the image is considered unwatermarked.

Let $\mathbf{c} = (c_1, ..., c_k)^T$ be the randlet transform coefficients of the image being considered. Let $\delta \in (0, \frac{\Delta}{2})$, and let $\tau \in [0, 1]$. Recall that coefficient c_j is quantized with quantizer Q_j. Define

$$F_\delta(x) = \begin{cases} 1 & \text{if } |x| \leq \delta \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Then an image is considered watermarked if

$$\frac{1}{k} \sum_{j=1}^{k} F_\delta(Q_j(c_j) - c_j) \geq \tau. \tag{4}$$

3.4.2 Verification Watermark, Soft Detection

The soft detector for verification watermarks is similar to the hard detector, but instead of using the function F_δ to reduce each coefficient to a 0 or 1, it estimates the probability that each coefficient is marked and uses those probabilities to decide if the image is watermarked or not.

Given $\mathbf{c} = (c_1, ..., c_k)^T$ as before, let $P_j : [0, \frac{\Delta}{2}] \to (0, 1)$ take as input the distance of coefficient c_j to the nearest quantization point on Q_j and return the probability that the coefficient was marked. P_j is determined experimentally by examining the effects of attacks on watermark coefficients. $P_j(x)$ is an estimate of the probability that coefficient c_j was marked if it was distance x from $Q_j(c_j)$. Note that even if a coefficient is on a

quantization point, there is a chance that c_j was not originally marked ($P_j(0) \neq 1$), and even if a coefficient is maximally far from a quantization point, there is a chance that c_j was originally watermarked ($P_j(\frac{\Delta}{2}) \neq 0$).

Let p_j be our estimate of the probability that c_j was marked, $p_j = P_j(Q_j(c_j) - c_j)$. The soft verification detector considers an image to be watermarked if the likelihood of being watermarked is sufficiently larger enough than the likelihood of not being watermarked. Letting $e^\tau \geq 1$ be a threshold constant, we write that the image is watermarked if:

$$\prod_{j=1}^{k} p_j > e^\tau \prod_{j=1}^{k} (1 - p_j),$$

$$\sum_{j=1}^{k} \ln \frac{p_j}{1 - p_j} > \tau. \tag{5}$$

3.4.3 Data Watermark, Hard Detection

With data watermarks, each randlet (and therefore coefficient) encodes a single bit. The goal is that the extracted data, $\mathbf{w} = (w_1, ..., w_k)$ is equal to the secret embedded data $\mathbf{v} = (v_1, ..., v_k)$. As described in section 3.2, there are two uniform quantizers for each coefficient, Q_j^0 to encode 0 bits and Q_j^1 to encode 1 bits.

$$Q_j^0 = \{..., -4\Delta + \alpha_j, -2\Delta + \alpha_j, \alpha_j, 2\Delta + \alpha_j, 4\Delta + \alpha_j, ...\},$$

$$Q_j^1 = \{..., -3\Delta + \alpha_j, -\Delta + \alpha_j, \Delta + \alpha_j, 3\Delta + \alpha_j, ...\}.$$

Note that the quantization points on Q_j^0 and Q_j^1 alternate, so any coefficient c_j will be between a quantization point on Q_j^0 and a quantization point on Q_j^1. With hard detection, coefficient c_j is assigned a value of 0 if it is closer to the quantization point on Q_j^0 and is assigned a value of 1 if it is closer to the quantization point on Q_j^1. Given $\mathbf{c} = (c_1, ..., c_k)^T$ as before, let

$$w_j = \begin{cases} 0 & \text{if } |Q_j^0(c_j) - c_j| < |Q_j^1(c_j) - c_j| \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

After each bit has been assigned a value, an error correcting code is used to further reduce bit errors.

3.4.4 Data Watermark, Soft Detection

Instead of assigning values to bits, the soft decoder calculates for each bit the probability that the bit is a 0 and the probability that it is a 1. An error correcting decoder then takes these probabilities and performs soft decoding to find the output bits. Error correcting codes must be used when performing soft detection. Error correcting codes can also be used to perform hard decoding after the soft decoding.

Given $\mathbf{c} = (c_1, ..., c_k)^T$ as before, let the embedded codeword be $\mathbf{v} = (v_1, ..., v_k)$. Similarly to soft decoding of verification watermarks, let $P_j : [0, \Delta] \to (0, 1)$ take as input the distance of coefficient c_j to the nearest quantization point and return the probability that the coefficient was originally quantized to that quantization point. As before, P_j is determined experimentally by examining the effects of attacks on watermark coefficients. Note that as before, $P_j(0) \neq 1$ and $P(\Delta) \neq 0$.

Figure 1: The upper-left shows the unwatermarked image, the upper-right shows the watermarked image, the lower-left shows the magnitude watermark, and the lower-right shows an equalized version of the watermark.

To perform soft detection, each bit is estimated to be a 0 with probability $P_j(Q_j^0(c_j) - c_j)$ and a 1 with probability $P_j(Q_j^1(c_j) - c_j)$, and these probabilities are passed to a soft decoder. As an example, we present a soft decoder for a repetition code.

Let $\mathbf{s} = (s_1, ..., s_n)$ be the secret data bits, and use a repetition rate of r, so $k = nr$. The embedded watermark is $\mathbf{v} = (v_1, ..., v_k)$, where $v_i = s_{\lceil \frac{i}{r} \rceil}$. The detected watermark is $\mathbf{w} = (w_1, ..., w_n)$. w_j is determined by examining coefficients $c_{(j-1)r+1}, ..., c_{jr}$.

Let p_j^0 and p_j^1 be our estimates of the probability that c_j was originally quantized to 0 and 1, respectively. Then $p_j^0 = P_j(Q_j^0(c_j) - c_j)$ and $p_j^1 = P_j(Q_j^1(c_j) - c_j)$. Comparing likelihoods, we estimate that $s_j = 0$ if

$$\prod_{i=(j-1)r+1}^{jr} p_j^0 > \prod_{i=(j-1)r+1}^{jr} p_j^1.$$

$$s_j = \begin{cases} 0 & \sum_{i=(j-1)r+1}^{jr} \ln p_j^0 - \ln p_j^1 > 0 \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

4 Experimental Results

Experimentally, attacks on an image produce very specific distributions of errors on the randlet transform coefficients. Desynchronization attacks, such as rotation and cropping, induce error distributions similar to a Laplacian distribution, while attacks that do not affect synchronization induce errors according to a normal distribution. Analysis of these error distributions and the effects on image identification, image hashing, and image watermarking is the subject of an upcoming paper.

Since the watermarks are fairly robust against desynchronization attacks, it is feasible to perform searches over the inverses of these attacks to find the watermark. For example, since the watermark extremely robust against rotations of 1.5 degrees, we have implemented a search which can successfully find the watermark at arbitrary rotations by incrementally rotating the watermarked image and testing for the watermark.

In our tests we found that small randlets were not as robust against attacks, while larger randlets begin to overlap too much, greatly limiting the number of randlets that

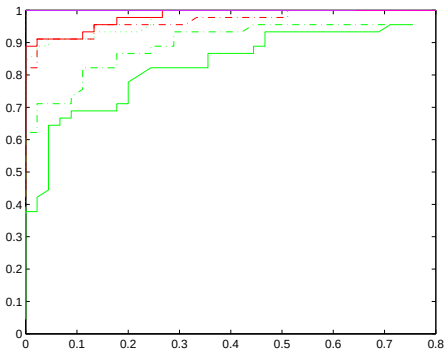

Figure 2: Performance for verification watermarking with soft detection. The y-axis is the probability of a true positive, the x-axis is the probability of a false positive. 15 images of size 512×512 were watermarked. 3 different randlet bases of 80×70 smooth random randlets were tested. 401 coefficients were embedded per image. Average distortion to signal ratio was 30db. Distortion compensation was used with $\alpha = 0.5$. Performance was perfect for 1.5 degrees rotation, 2% cropping, additive Gaussian noise of variance 0.25 (pixel values in [0,1]), and JPEG compression with quality 10. The dashed blue line corresponds to 2 degrees of rotation and the solid blue line corresponds to 2.5 degrees of rotation. The dashed red line corresponds to 4% cropping and the solid red line corresponds to 6% cropping.

could be used. The frequencies present in the randlets were also important. Randlets with many high-frequency components were not be as robust against desynchronization attacks (for example, random randlets), and randlets with only low-frequency components raised the condition number of $\mathbf{S}$ and greatly increased embedding distortion. Soft detection was found to be much more robust than hard detection, as expected.

References

[1] E. Candès and D. Donoho. Curvelets: A surprisingly effective nonadaptive representation of objects with edges. Technical report, 1999.

[2] B. Chen and G. W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Trans. on Information Theory*, 47(4):1423–1443, May 2001.

[3] J. Fridrich. Key-dependent random image transforms and their applications in image watermarking. *International Conference on Imaging Science, Systems, and Technology*, pages pp. 237–243, 1999.

[4] J. Fridrich. Visual hash for oblivious watermarking. *Proc. SPIE Photonic West Electronic Imaging*, 2000.

[5] R. V. M. K. Mihcak and M. Kesal. Watermarking via optimization algorithms for quantizing randomized statistics of image regions. *Proceedings of the 40th annual Allerton Conference on Communications, Computing and Control*, 2002.

[6] M. Malkin and R. Venkatessan. The randlet transform: Applications to universal perceptual hashing and image identification. *42nd Annual Allerton Conference on Communications, Control, and Computing*, 2004.

[7] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[8] M. K. Mihcak and R. Venkatesan. Blind image watermarking via derivation and quantization of robust semi–global statistics. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2002.