Digital Watermarking

Melinos Averkiou

1. Introduction

Digital watermarking is the act of hiding a message related to a digital signal (i.e. an image, song, video) within the signal itself. It is a concept closely related to steganography, in that they both hide a message inside a digital signal. However, what separates them is their goal. Watermarking tries to hide a message related to the actual content of the digital signal, while in steganography the digital signal has no relation to the message, and it is merely used as a cover to hide its existence.
Watermarking has been around for several centuries, in the form of watermarks found initially in plain paper and subsequently in paper bills. However, the field of digital watermarking was only developed during the last 15 years and it is now being used for many different applications.
In the following sections I will present some of the most important applications of digital watermarking, explain some key properties that are desirable in a watermarking system, and give an overview of the most common models of watermarking as presented in the book by Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Friedrich and Ton Kalker [1]. These basic models will be further illustrated by the use of example watermarking systems that were developed in Matlab. All images used in this essay, except those used to present the results of the example watermarking systems are taken from this book [1].

2. Watermarking applications

The increasing amount of research on watermarking over the past decade has been largely driven by its important applications in digital copyrights management and protection.
One of the first applications for watermarking was broadcast monitoring. It is often crucially important that we are able to track when a specific video is being broadcast by a TV station. This is important to advertising agencies that want to ensure that their commercials are getting the air time they paid for. Watermarking can be used for this purpose. Information used to identify individual videos could be embedded in the videos themselves using watermarking, making broadcast monitoring easier.
Another very important application is owner identification. Being able to identify the owner of a specific digital work of art, such as a video or image can be quite difficult. Nevertheless, it is a very important task, especially in cases related to copyright infringement. So, instead of including copyright notices with every image or song, we could use watermarking to embed the copyright in the image or the song itself.
Transaction tracking is another interesting application of watermarking. In this case the watermark embedded in a digital work can be used to record one or more transactions taking place in the history of a copy of this work. For example, watermarking could be used to record the recipient of every legal copy of a movie by embedding a different watermark in each copy. If the movie is then leaked to the Internet, the movie producers could identify which recipient of the movie was the source of the leak.

Finally, copy control is a very promising application for watermarking. In this application, watermarking can be used to prevent the illegal copying of songs, images of movies, by embedding a watermark in them that would instruct a watermarking-compatible DVD or CD writer to not write the song or movie because it is an illegal copy.

3. Watermarking properties

Every watermarking system has some very important desirable properties. Some of these properties are often conflicting and we are often forced to accept some trade-offs between these properties depending on the application of the watermarking system.

The first and perhaps most important property is effectiveness. This is the probability that the message in a watermarked image will be correctly detected. We ideally need this probability to be 1.

Another important property is the image fidelity. Watermarking is a process that alters an original image to add a message to it, therefore it inevitably affects the image's quality. We want to keep this degradation of the image's quality to a minimum, so no obvious difference in the image's fidelity can be noticed.

The third property is the payload size. Every watermarked work is used to carry a message. The size of this message is often important as many systems require a relatively big payload to be embedded in a cover work. There are of course applications that only need a single bit to be embedded.

The false positive rate is also very important to watermarking systems. This is the number of digital works that are identified to have a watermark embedded when in fact they have no watermark embedded. This should be kept very low for watermarking systems.

Lastly, robustness is crucial for most watermarking systems. There are many cases in which a watermarked work is altered during its lifetime, either by transmission over a lossy channel or several malicious attacks that try to remove the watermark or make it undetectable. A robust watermark should be able to withstand additive Gaussian noise, compression, printing and scanning, rotation, scaling, cropping and many other operations.

4. Watermarking models

There are several ways in which we can model a watermarking process. These can be broadly classified in one of two groups. The first group contains models which are based on a communication-based view of watermarking and the second group contains models based on a geometric view of watermarking. In the rest of this essay, I only refer to image watermarking because I only concentrated on images during the development of example watermarking systems.

4.1 Communication-based models

Communication-based models describe watermarking in a way very similar to the traditional models of communication systems. Watermarking is in fact a process of communicating a message from the watermarking embedder to the watermarking receiver. Therefore, it makes sense to use the models of secure communication to model this process.

In a general secure communication model we would have the sender on one side, which would encode a message using some kind of encoding key to prevent eavesdroppers to decode the message if the message was intercepted during transmission. Then the message would be transmitted on a communications channel, which would add some noise to the noise to the encoded message. The resulting noisy message would be received at the other end of the transmission by the receiver, which would try to decode it using a decoding key, to get the original message back. This process can be seen in Figure 1.

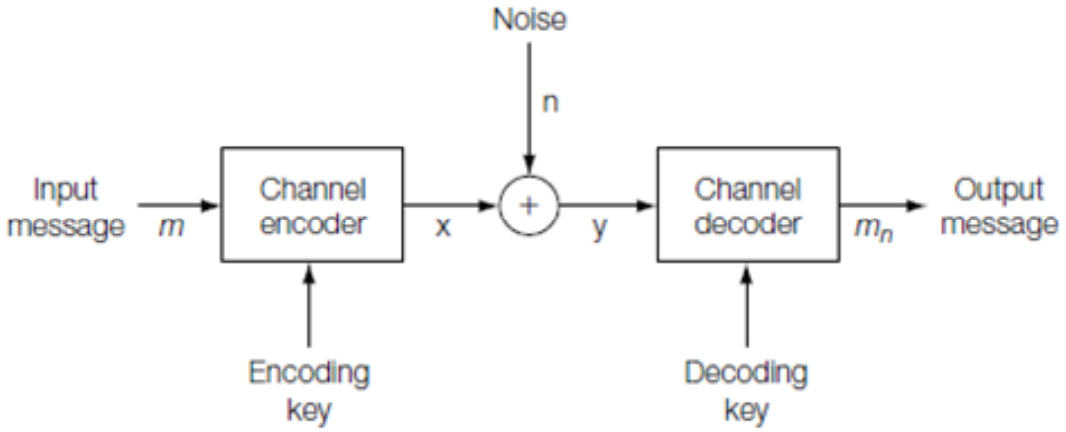

Figure 1 Standard model of a communications channel with key-based encoding

In general, communication-based watermarking models can be further divided into two sub-categories. The first uses side-information to enhance the process of watermarking and the second does not use side-information at all. The term side-information refers to any auxiliary information except the input message itself, that can be used to better encode or decode it. The best example of this is the image used to carry the message, which can be used to provide useful information to enhance the correct detection of the message at the receiver.

4.2 Geometric models

It is often useful to think of watermarking in geometric terms. In this type of model, images, watermarked and unwatermarked, can be viewed as high-dimensional vectors, in what is called the media space. This is also a high-dimensional space that contains all possible images of all dimensions. For example a 512 X 512 image would be described as a 262144 elements vector in a 262144-dimensional space.
Geometric models can be very useful to better visualize the watermarking process using a number of regions based on the desirable properties of watermarking. One of these regions is the embedding region, which is the region that contains all the possible images resulting from the embedding of a message inside an unwatermarked image using some watermark embedding algorithm. Another very important region is the detection region, which is the region containing all the possible images from which a watermark can be successfully extracted using a watermark detection algorithm. Lastly, the region of acceptable fidelity contains all the possible images resulting from the embedding of a message into an unwatermarked image, which essentially look identical to the original image. The embedding region for a given watermarking system should ideally lie inside the intersection of the detection region

and the region of acceptable fidelity, in order to produce successfully detected watermarks that do not alter the image quality very much.
An example of a geometric model can be seen in Figure 2. Here we can see that if mean square error (MSE) is used as a measure of fidelity, the region of acceptable fidelity would be an n-dimensional sphere centred on the original unwatermarked image (c_o), with a radius defined by the largest MSE we are willing to accept for images with acceptable fidelity. The detection region for a detection algorithm based on linear correlation would be defined as a half space, based on the threshold used to decide whether an image has a watermark embedded or not. Note that the diagram is merely a projection of an n-dimensional space into a 2d space.

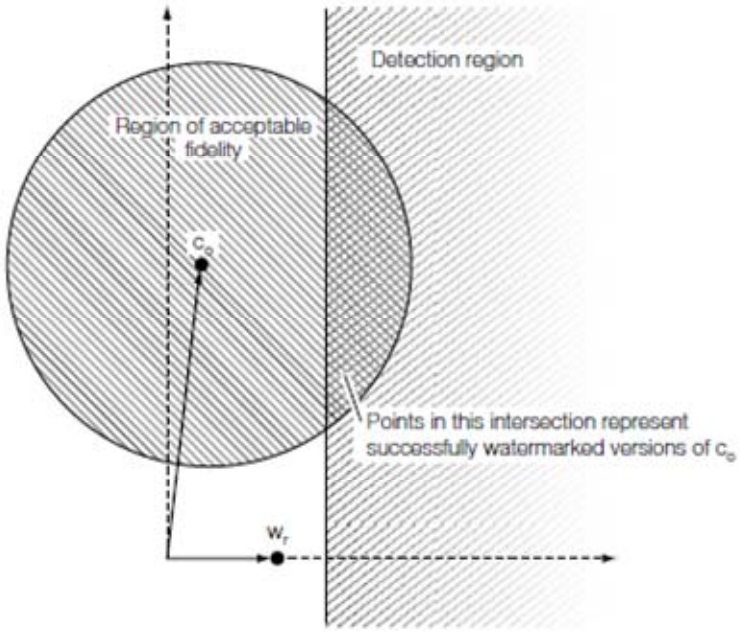

Figure 2 The region of acceptable fidelity (defined by MSE) and the detection region (defined by linear correlation)

When thinking about complex watermarking systems, it is sometimes more useful to consider a projection of the media space into a possibly lower-dimension marking space in which the watermarking then takes place as usual. This projection can be handled more easily by computers because of the smaller number of vector elements and can be possibly expressed by block-based watermarking algorithms which separate images into blocks instead of operating on a pixel basis.

5. Watermarking without side-information

As described earlier, some communication-based watermarking models do not take advantage of the channel side-information. In this kind of models, the image is simply considered as another form of channel noise that distorts the message during its transmission. This can be seen in Figure 3. The watermark embedder encodes a message using a watermark encoder and a key. This is then added to the original image and transmitted over the communication channel which adds some noise. The watermark detector at the other end receives the noisy watermarked image and tries to decode the original image using a key.

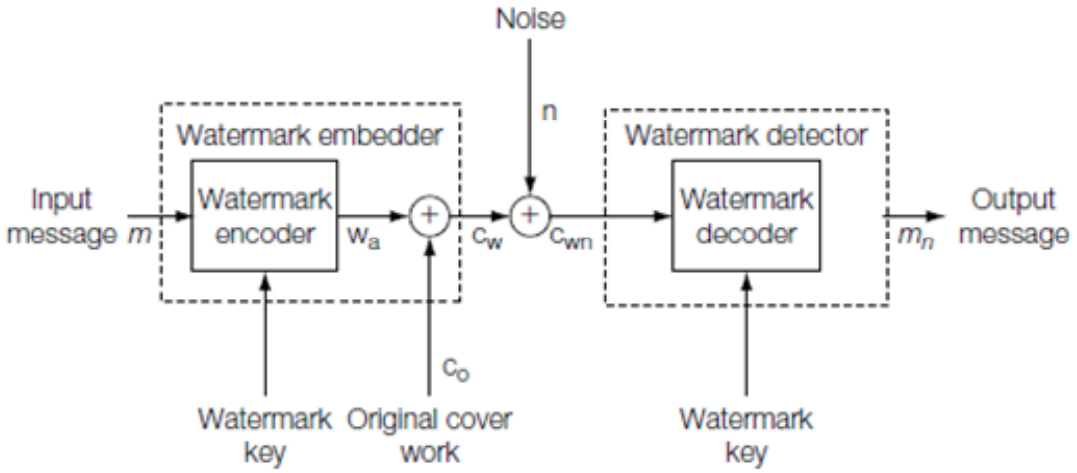

Figure 3 Standard model for watermarking with no side-information

An example watermarking system that illustrates watermarking without side-information was implemented in Matlab.

5.1 Blind embedding and linear correlation detection

This system is an example of blind embedding, which does not exploit the original image statistics to embed a message in an image. The detection is done using linear correlation. This system is a 1-bit watermarking system, in other words it only embeds one bit (a 1 or 0) inside the cover image. The algorithm for the embedder and the detector is as follows:

Embedder:

1. Choose a random reference pattern. This is simply an array with the same dimensions as the original image, whose elements are drawn from a random Gaussian distribution in the interval [-1, 1]. The watermarking key is the seed that is used to initiate the pseudo-random number generator that creates the random reference pattern.
2. Calculate a message pattern depending on whether we are embedding a 1 or a 0. For a 1, leave the random reference pattern as it is. For a 0, take its negative to get the message pattern.
3. Scale the message pattern by a constant α which is used to control the embedding strength. For higher values of α we have more robust embedding, at the expense of loosing image quality. The value used at the initial experiment was $\alpha = 1$.
4. Add the scaled message pattern to the original image to get the watermarked image.

Detector:

1. Calculate the linear correlation between the watermarked image that was received and the initial reference pattern that can be recreated using the initial seed which acted as the watermarking key.
2. Decide what the watermark message was, according to the result of the correlation. If the linear correlation value was above a threshold, we say that the message was a 1. If the linear correlation was below the negative of the

threshold we say that the message was a 0. If the linear correlation was between the negative and the positive threshold we say that no message was embedded.

An example of the embedding process can be seen in Figure 4. The top left image is the original image, the bottom left image is the reference pattern and the watermarked image resulting from embedding a 1, with α=1, is seen on the right. As we can see, there is no perceptual difference between the original and the watermarked image.

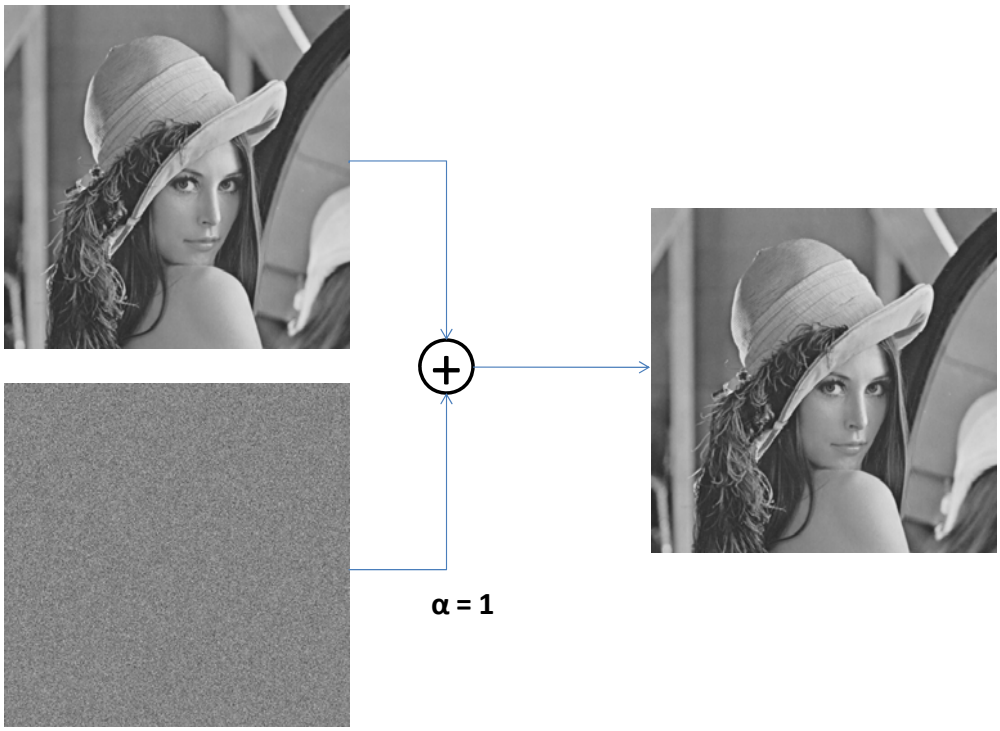

Figure 4 An example of the blind embedding system results

To test the effectiveness of this system, I used 400 small images (112 X 92 pixels). I ran the embedding algorithm with α = 1 and tried to embed a 1 and a 0 in each of these images, resulting in 800 watermarked images. I then calculated the linear correlation (as in the detector) for each one of those images, as well as for the original, unwatermarked images and plotted the result. This can be seen in Figure 5. The x-axis is the linear correlation value, while the y axis is the percentage of images that had a specific linear correlation value. As we can see, most of the images that had a 0 embedded had a value centred on -1. Those that had a 1 embedded had a value centred on 1, while unwatermarked images had a value centred on 0. The problem is that the three graphs overlap at points between -1 and 0, as well as between 0 and 1. This means that choosing a specific threshold can be difficult because no matter what threshold we chose, there will always be some unwatermarked images classified as having a watermark (false positives) while some others that had a watermark will be classified as unwatermarked (false negatives). Ideally we want these three graphs to be non-overlapping and far from each other so we can choose a threshold that gives no false positives or false negatives.

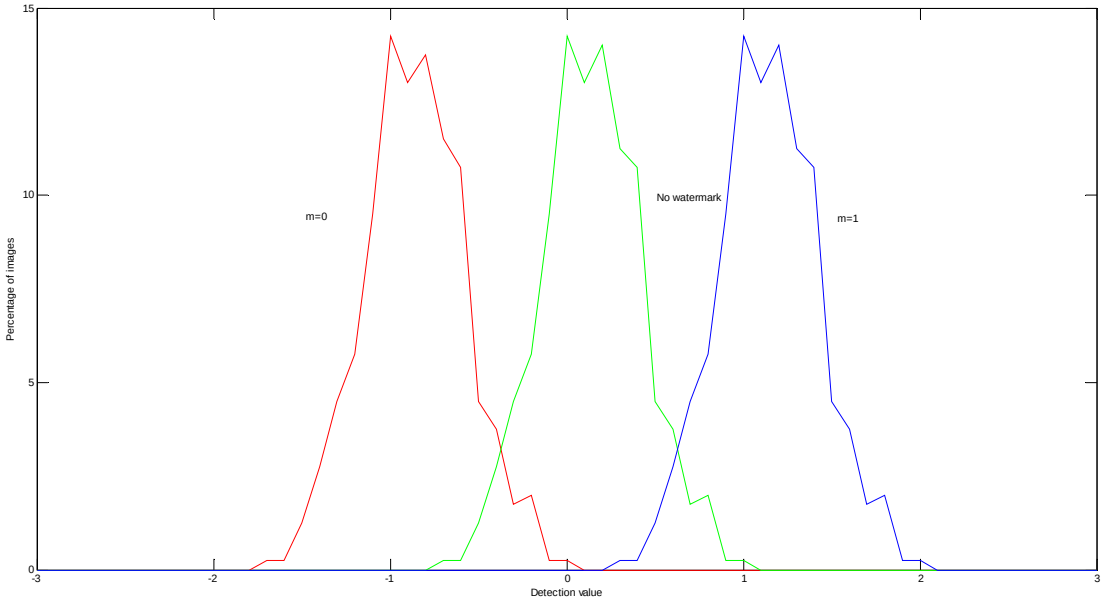

Figure 5 Detection values for watermarked images

The reference pattern is very important for any watermarking system, but especially for this kind of system that does not use any side-information in the original image. In order to illustrate this, I ran the same test as above, only this time I applied a low-pass 5 X 5 averaging filter on the reference pattern before adding it to each image. The results can be seen in Figure 6. The x-axis is the linear correlation value, while the y axis is the percentage of images that had a specific linear correlation value. As we can see, the overlapping of the three graphs representing the percentage of images as a function of the linear correlation value for the three groups of images (1 embedded, 0 embedded, no watermark) is much more obvious that the original experiment, and it in practise impossible to choose an appropriate threshold that does not result in a very high false positive rate.

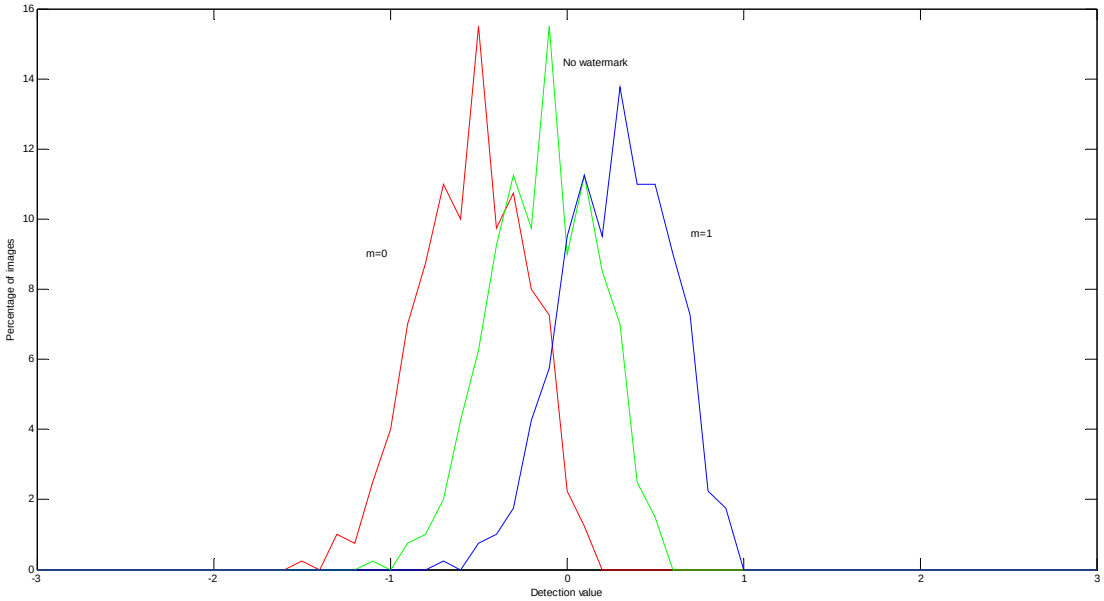

Figure 6 Detection values for watermarked images after applying a low-pass filter to the reference pattern

The value of α is also crucial for this simple system, as it can increase the embedding strength by essentially moving the linear correlation values further apart so we have less overlap between linear correlation graphs for the three image groups. To test this fact, I ran the original 400 image test with a value of 2 for α. The results were more impressive this time, as we can see in Figure 7.

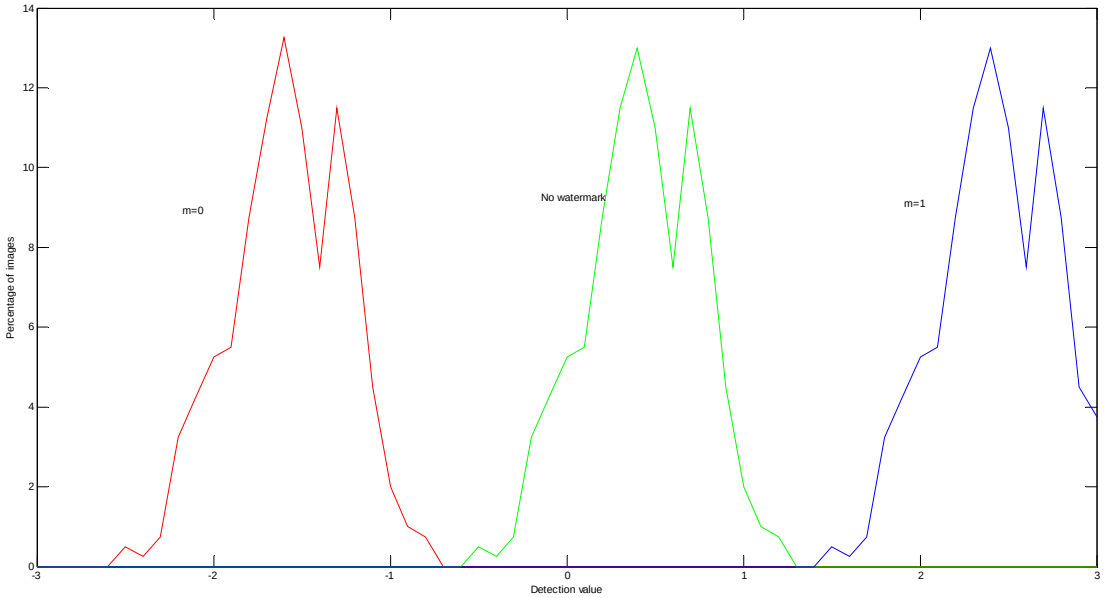

Figure 7 Detection values for watermarked images with α = 2

The linear correlation graph for images with a 0 embedded moves towards -2, while the graph for images with a 1 embedded moves towards 2. The unwatermarked images graph is between 0 and 1. Here there is a clear separation between the three graphs and no overlapping sections exist. This means we can confidently choose a threshold for classifying watermarked and unwatermarked images.

In order to test the robustness of the system, I also ran a modified version of the original 400 images test, in which I added random Gaussian noise in the interval [-10,10] to each pixel of the watermarked image, to see how that would affect the watermark detection. The results can be seen in Figure 8. As it is expected, the three graphs move closer together than the original experiment. This makes their overlapping regions bigger and in turn that means threshold selection becomes more difficult. The overall conclusion is that this system is not resilient to additive noise.

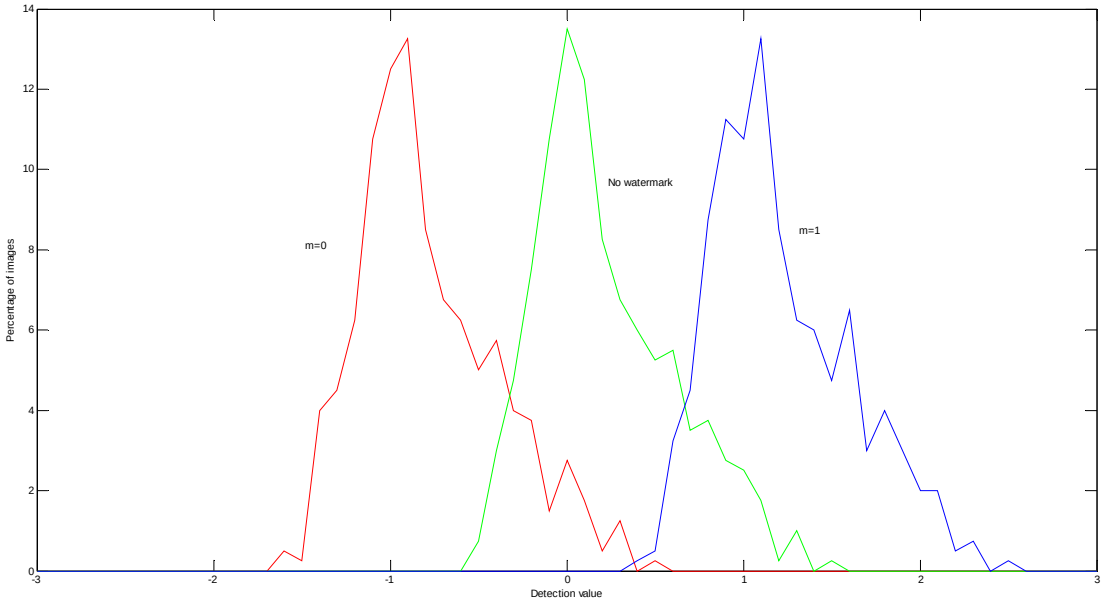

Figure 8 Detection values for watermarked images after adding noise to them

6. Watermarking with side-information

Some of the weaknesses of systems that do not exploit side-information were identified by the blind embedding example system. The robustness and effectiveness of the watermarking process can be greatly increased by taking advantage of the original image information during the embedding stage. The general form of watermarking models that exploit side-information is shown in Figure 9.

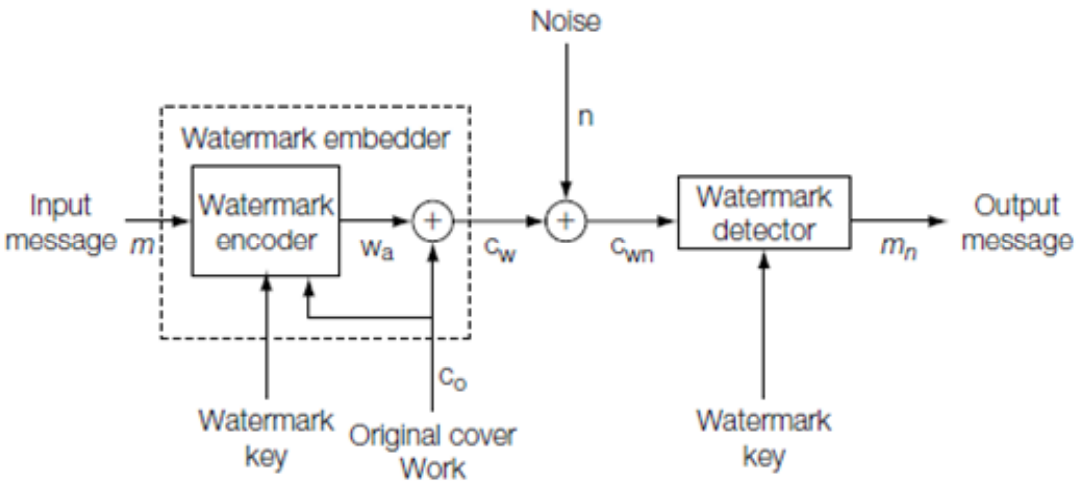

Figure 9 Standard model for watermarking with side-information

The only difference between this model and the model of section 2 is the use of the original image. The watermark embedder still encodes a message using not only a key but also the information provided by the original image. The resulting encoded message is then added to the original image as in the case of the no-side-information model. The noisy transmission channel adds some more noise, and the watermarking detector tries to get the original message back using the original key and a detection algorithm.

There were three examples implemented to illustrate the use of watermarking with the use of side-information. The first is a simple extension of the blind embedding example, while the other two are based on dirty-paper coding, a class of popular watermarking algorithms.

6.1 Informed embedding and linear correlation detection

This system is essentially an extension of the blind embedding system, which aims to provide 100% effectiveness in the watermarking detection, by controlling the value of the α parameter that has been proved to be extremely important in the experiments conducted. The system is also a 1-bit watermarking system. The algorithm for the embedder and the detector is the following:

Embedder:

1. Choose a random reference pattern. This is simply an array with the same dimensions as the original image, whose elements are drawn from a random Gaussian distribution in the interval [-1, 1]. The watermarking key is the seed that is used to initiate the pseudo-random number generator that creates the random reference pattern.
2. Calculate a message pattern depending on whether we are embedding a 1 or a 0. For a 1, leave the random reference pattern as it is. For a 0, take its negative to get the message pattern.
3. Calculate the α constant using the linear correlation between the watermarked image and the message pattern. The goal is to ensure that the linear correlation value used for the watermark detection is always some constant, greater than the detection threshold. We substitute this value in the left hand side of the linear correlation equation and solve for α. In this way we get a different value for α for every image.
4. Scale the message pattern by α which is used to control the embedding strength.
5. Add the scaled message pattern to the original image to get the watermarked image.

Detector:

1. Calculate the linear correlation between the watermarked image that was received and the initial reference pattern that can be recreated using the initial seed which acted as the watermarking key.
2. Decide what the watermark message was, according to the result of the correlation. If the linear correlation value was above a threshold, we say that the message was a 1. If the linear correlation was below the negative of the threshold we say that the message was a 0. If the linear correlation was between the negative and the positive threshold we say that no message was embedded.

An example of the embedding process can be seen in Figure 10. The top left image is the original image, the bottom left image is the reference pattern and the watermarked image resulting from embedding a 1 is seen on the right. As we can see, there is no perceptual difference between the original and the watermarked image.

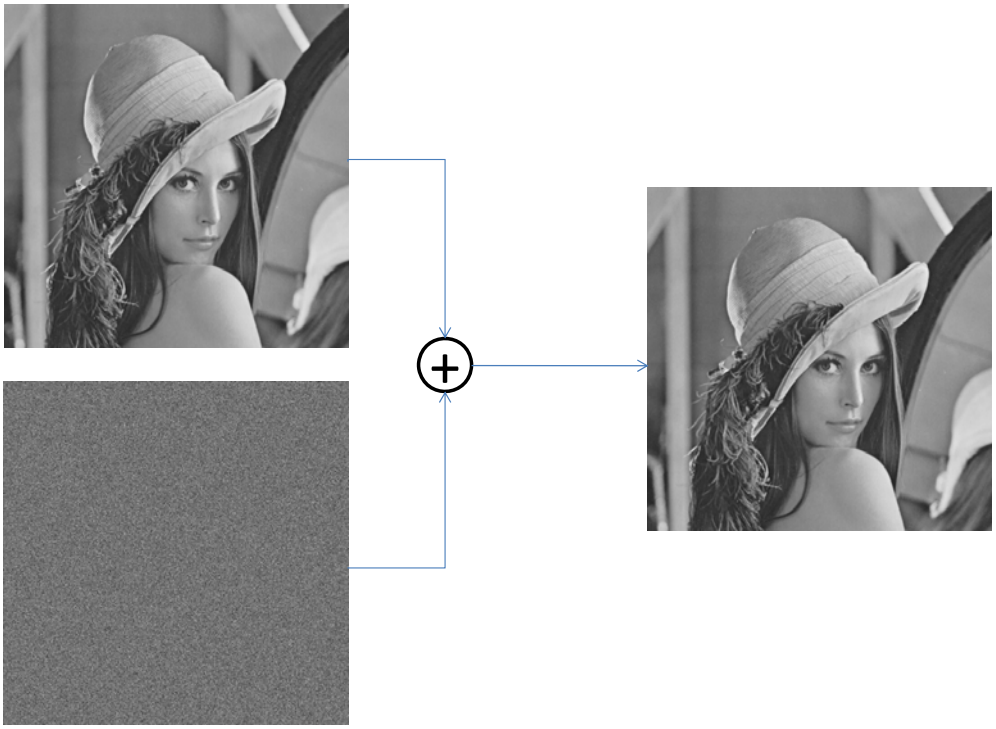

Figure 10 An example of the informed embedding system results

To test the effectiveness of this system, I used 400 small images (112 X 92 pixels). I ran the embedding algorithm with a threshold of 0.7 and β= 0.3 (β is added to be confidently above the threshold) to embed a 1 and a 0 in each of these images, resulting in 800 watermarked images. I then calculated the linear correlation (as in the detector) for each one of those images, as well as for the original, unwatermarked images and plotted the result. This can be seen in Figure 11. The x-axis is the linear correlation value, while the y axis is the percentage of images that had a specific linear correlation value.

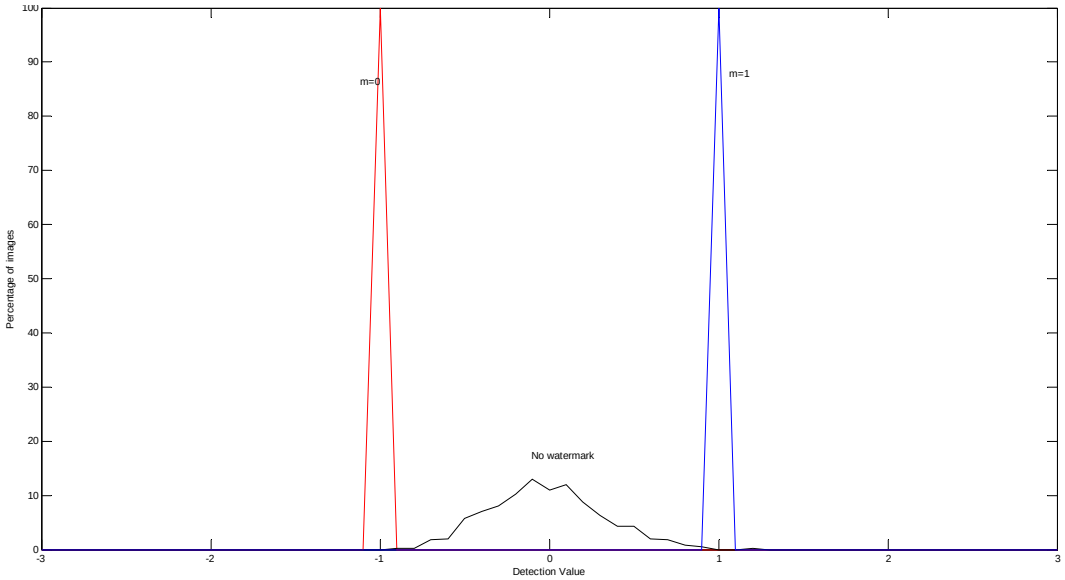

Figure 11 Detection values for watermarked images

It is obvious from the figure that the three graphs representing images that had either a 1 or a 0 or no watermark embedded are well separated and there are no overlaps between them. This is due to the fact that the value of α is dynamically adjusted to create watermarked images with a detection value of 1.0 (0.7 for the threshold plus 0.3 for β). Therefore, images that had a 1 embedded have a detection value of 1 and those that had a 0 embedded have a detection value of -1. The original images' detection values are centred on 0 and this graph is essentially the same as the one created by the original experiment in the blind embedding system. The reason why the watermarked images do not all have detection values of -1 and 1, as would be expected by this fixed-detection value algorithm has to do with numerical errors introduced during round-off and truncation of values for converting them to integers. It is evident that this system has 100% effectiveness and a threshold can be chosen to provide watermarking detection with a zero false positive rate.
Even in the presence of additive noise, the system is quite robust. I modified the above experiment to add Gaussian noise in the interval [-10,10] to every pixel of the watermarked images, in order to see the changes in the three graphs. The results are shown in Figure 12.

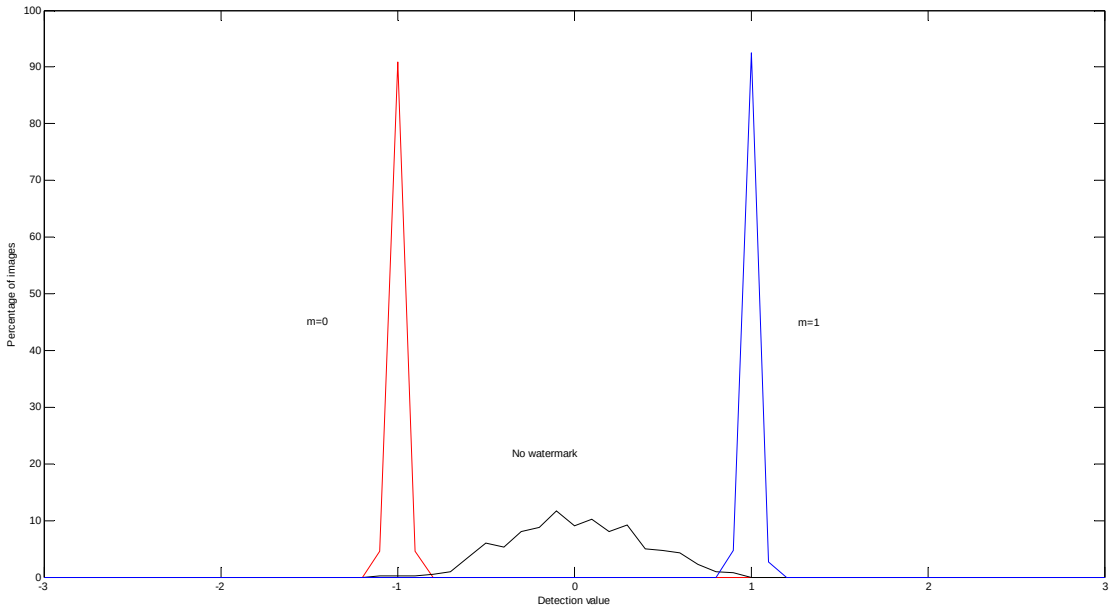

Figure 12 Detection values for watermarked images after adding noise to them

There is essentially no difference between Figure 12 and Figure 11, which indicates that the presence of noise did not affect the detection values much. Therefore, we can conclude that the system is quite robust to additive noise, which was not the case for the blind embedding system. Overall, the informed embedding system performs better, both in terms of effectiveness and in terms of robustness, than the blind embedding system.

7. Dirty-paper codes

A dirty-paper code is a collection of codes. In contrast to classical message coding, where each message is represented by a single code word, in a dirty paper code each message is represented by a set of code words. In order to transmit a message, the encoder needs to find the code word that best fits the signal into which the message is to be embedded. Therefore, for every message we have a set of code words, which are

called sub-code books. The union of these sub-code books is the dirty paper code. The decoder at the other end of the transmission needs to find the code word closest to the signal that is received. The sub-code which contains this code word is then used to classify the message that was originally embedded in the received signal.
The term dirty paper was coined in Max Costa's seminal paper "Writing on Dirty Paper" [2] in which he used the dirty paper analogy. The main idea is that when writing a symbol on a dirty sheet of paper, this is best done by adapting the symbol to the dirt spots which are already on the paper sheet. This is illustrated in Figure 13, for the example of writing the symbol A on a dirty paper with blind and informed writing. In watermarking, the dirty paper is the cover image which is known to the embedder but not to the detector. Using multiple codes for every message, we can select the best code for a specific image.

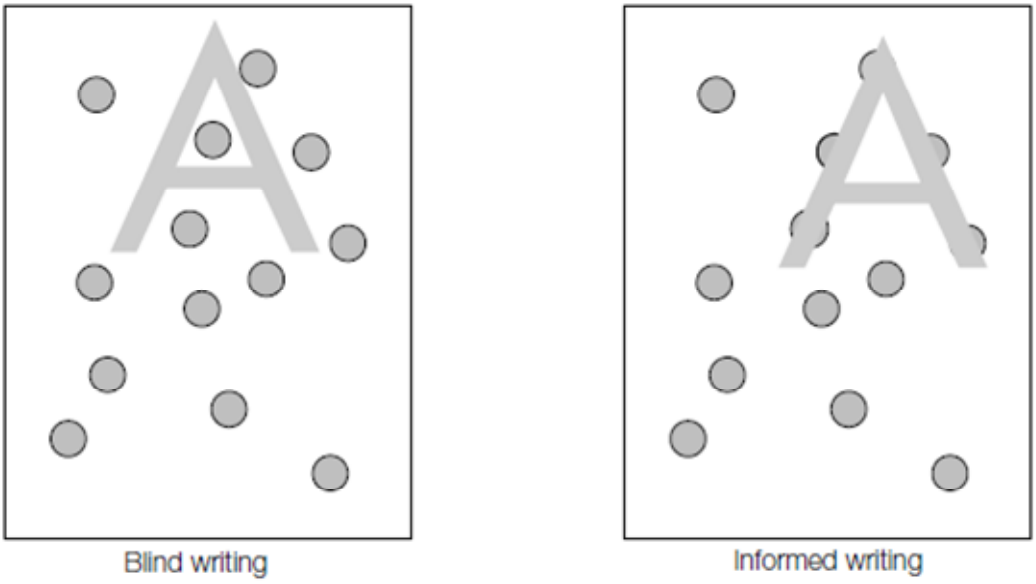

Figure 13 Writing A on a dirty paper using blind and informed writing

The quality of dirty paper codes is based on the trade-off between the density of the sub-code books and the density of the union of these sub-code books. If we have dense sub-code books, it is more likely that we will find a code word closer to the original image. However, this means that the union of the sub-code books will also be dense, which in turn implies that it will be easier to make decoding errors due to the different sub-code books being closer. Ideally we would like to have dense sub-code books to provide better image fidelity and sparse unions of sub-code books to provide better robustness. These are contradictory requirements and we need to find a balance between them.

7.1 Block-based/fixed robustness embedding-correlation coefficient detection

This system is a relatively simple application of dirty paper codes for embedding a single bit (either 1 or 0) into an image. It uses block-based embedding, which means that the watermarking process is done on smaller dimensionality vectors extracted by using blocks instead of pixels in the image. It also uses fixed robustness embedding, which means that the maximum amount of noise the watermark embedded into the cover image can withstand is kept constant rather than keeping the correlation value constant as in the informed embedding system. The detector uses the correlation

coefficient instead of linear correlation to decide what message was embedded into an image. The algorithm for the embedder and detector of this system follows:
Embedder:

1. Create two sets of 8 x 8 reference patterns, one corresponding to a message of 1 and one corresponding to a message of 0. The size of the two sets is the same and it is crucial to preserving the image quality.
2. Extract a 64-element vector from the original image by summing 8 x 8 blocks of the image and then taking the average of the resulting 8 x 8 sum block.
3. Find the correlation coefficient of the vector extracted from the original image and each of the reference patterns in the set corresponding to the message we want to embed. If we want to embed a 1, we only check the reference patterns created for 1, not those created for 0.
4. Choose the reference pattern with the highest correlation coefficient (most similar to the image) and embed it into the image using a fixed robustness algorithm and reprojecting it into the same dimensions as the original image. The fixed robustness algorithm is not relevant in this discussion and is therefore not described.

Detector:

1. Extract a 64-element vector from the watermarked image by summing 8 x 8 blocks of the image and then taking the average of the resulting 8 x 8 sum block.
2. Find the correlation coefficient of the vector extracted from the watermarked image and each of the reference patterns in the two sets corresponding to 1 and 0.
3. The reference pattern with the highest correlation coefficient then classifies the watermark message. If the correlation coefficient of this reference pattern is below a give threshold, then the image has no watermark embedded. If this reference pattern belongs to the set of reference patterns corresponding to 1 and its correlation coefficient is above the given threshold, then the original message was a 1. If this reference pattern belongs to the set of reference patterns corresponding to 0 and its correlation coefficient is above the given threshold, then the original message was a 0.

An example of the result of the embedder can be seen in Figure 14.

Figure 14 An example of the dirty-paper coding system results

Unfortunately the image quality is degraded by the embedding process, and this is visually obvious if we compare the original and the watermarked image. In order to investigate the effect of the size of the reference pattern sets on the fidelity of the watermarked image I ran a test in which I gradually increased the number of reference patterns in each set. I then calculated the MSE between the original image and the resulting watermarked image for 400 small images (112 x 92 pixels). I embedded a 1 as well as a 0 in each of these images, resulting in 800 watermarked images and consequently 800 MSE values for a given reference pattern set size. Then I plotted the average of this MSE value for reference pattern set sizes ranging from 1 to 40. The resulting graph can be seen in Figure 15.

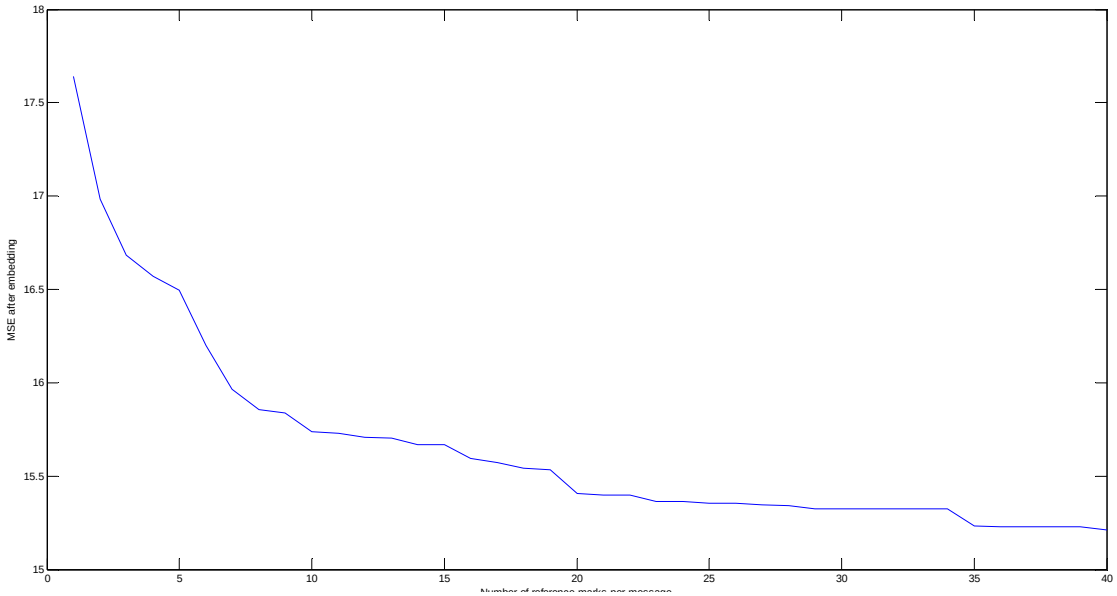

Figure 15 Average MSE for watermarked images with respect to the reference pattern set size

The x axis represents the number of reference patterns for each run of the experiment, ranging from 1 to 40. The y axis is the average MSE resulting from each run of the experiment. As we can see, although the MSE starts at a quite high value of 17.5 for only one reference pattern, it gradually falls to less than 15.5 for 40 reference patterns. This indicates the fact that using more reference patterns to encode a message results in increased fidelity because there is a higher chance to find a reference pattern closer to the original image.

The effectiveness of this system was also investigated in the same way as in the previous two systems. I used 400 small images (112 x 92 pixels). I ran the embedding algorithm with a reference pattern set size of 40, to embed a 1 and a 0 in each of these images, resulting in 800 watermarked images. I then calculated the correlation coefficient (as in the detector) for each one of those images and plotted the result. This can be seen in Figure 16. The x-axis is the linear correlation value, while the y axis is the number of images that had a specific correlation coefficient value. The red graph represents the images that had a 0 embedded and the blue graph represents the images that had a 1 embedded. A we can see, the correlation coefficient is above 0.99 in both graphs, which means that watermarks can be detected with a 100% effectiveness.

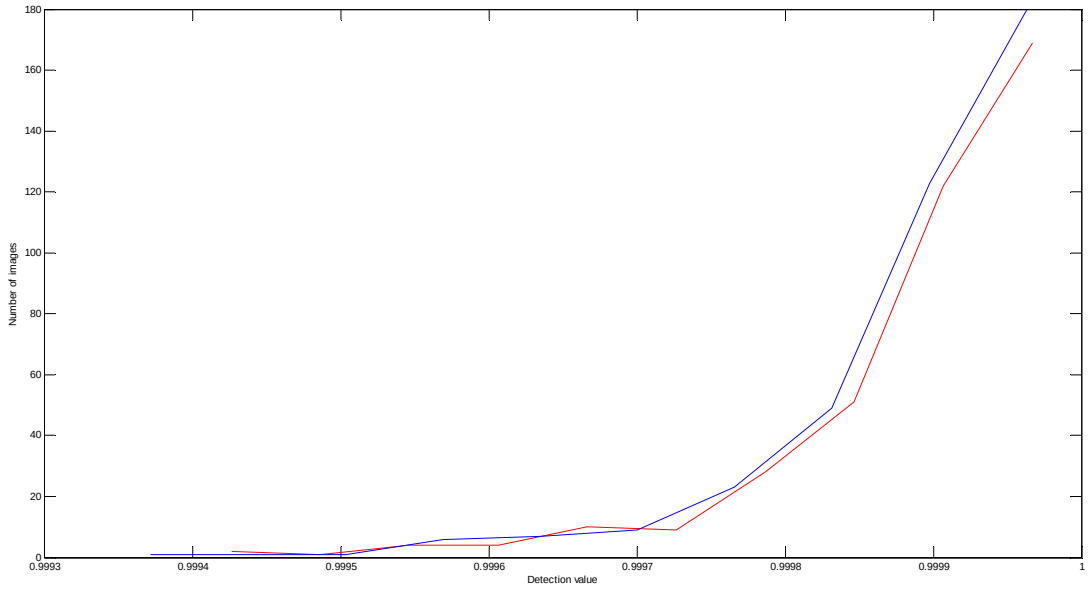

Figure 16 Detection values for watermarked images

8. Conclusion

Watermarking is a very active research field with a lot of applications. Although it is a relatively new field, it has produced important algorithms for hiding messages into digital signals. These can be described by many different models. Two broad categories for these models were described in this essay. These are communication-based models and geometric models. Communication-based models can be further divided into those which use side-information and those that don't. One example system was used to illustrate non-side-information models, and two example systems were used to illustrate side-information models. Each of these systems has its advantages and disadvantages, and each one trades some important watermarking property for another. The choice of which to use relies on the underlying application's requirements.

Of course the examples provided in this essay are only a small sample of the many different approaches to watermarking. Examples of other approaches that have not been mentioned include those which operate in the frequency domain and take advantage of DCT coefficient and wavelet coefficients.

References

[1] Cox I, Miller M, Bloom J, Fridrich J, Kalker T (2008) Digital Watermarking and Steganography Second Edition. Elsevier, 2008

[2] Costa M (1983) Writing on dirty paper. IEEE Transactions in Information Theory, 29:439–441, 1983.

APPENDIX A

A.1 Matlab code for the blind embedding and linear correlation detection system

```
%Blind embedding and linear correlation detection
clear all;
%Generate a reference pattern the same size as the images
wr=randn(112,92);
tm=mean2(wr);
wr=wr-tm;%zero mean
ts=std2(wr);
wr=wr/ts;%unit variance

%Show the reference pattern
figure
imshow(wr,[]);

%400 face images, 112 x 92 pixels, in 40 folders of 10 images each
numclass=40;
numface=10;

for s=1:numclass
    for i=1:numface
        %Read image
        c0=double(imread(strcat('C:\facedb_bmp\s',num2str(s),'\',
        num2str(i),'.bmp')));
        %Add reference pattern to the image to embed a 1
        wa=double(c0+wr);
        %Add negative reference pattern to the image to embed a 0
        nwr=wr*-1;
        nwa=double(c0+nwr);
        %Calculate linear correlation for images carrying a 1, a 0,
        %or no watermark and store it into a vector
        corr1(10*(s-1)+i)=sum(sum(wa.*wr))/(112*92);
        corr0(10*(s-1)+i)=sum(sum(nwa.*wr))/(112*92);
        corrN(10*(s-1)+i)=sum(sum(c0.*wr))/(112*92);
    end
end
%Calculate the histograms for the detection-value vectors
[a1,b1]=hist(corr1,-3:0.1:3);
[a2,b2]=hist(corr0,-3:0.1:3);
[a3,b3]=hist(corrN,-3:0.1:3);
%Plot the histograms
figure
plot(b2,a2/400*100,'red');
gtext('m=0');
hold;
plot(b3,a3/400*100,'green');
gtext('No watermark');
plot(b1,a1/400*100,'blue');
gtext('m=1');
xlabel('Detection value');
ylabel('Percentage of images');
```

A.2 Matlab code for the informed embedding and linear correlation detection system

```
%Informed embedding and linear correlation detection
clear all
%Generate a reference pattern the same size as the images
wr=randn(112,92);
tm=mean2(wr);
wr=wr-tm;%zero mean
ts=std2(wr);
wr=wr/ts;%unit variance

%Show the reference pattern
figure
imshow(wr,[]);

%400 face images, 112 x 92 pixels, in 40 folders of 10 images each
numclass=40;
numface=10;

for s=1:numclass
    for i=1:numface
        %Read image
        c0=double(imread(strcat('C:\facedb_bmp\s',num2str(s),'\',
        num2str(i),'.bmp')));
        %Find the image size
        [x,y]=size(c0);
        %Find alpha
        beta=1;
        alpha1=(x*y*beta-sum(sum(c0.*wr)))/(sum(sum(wr.*wr)));
        wrn=wr*(-1);
        alpha2=(x*y*beta-sum(sum(c0.*wrn)))/(sum(sum(wrn.*wrn)));
        wa1=(alpha1.*wr);
        wa2=(alpha2.*wr);
        %Embed a 1 and store the linear correlation
        cw1=double(c0+wa1);
        corr1(10*(s-1)+i)=sum(sum(cw1.*wr))/(x*y);
        %Embed a 0 and store the linear correlation
        cw2=double(c0-wa2);
        corr0(10*(s-1)+i)=sum(sum(cw2.*wr))/(x*y);
        %Also store the linear correlation for non-watermarked images
        corrN(10*(s-1)+i)= sum(sum(cw3.*wr))/(x*y);
     end
end
%Calculate the histograms for the detection-value vectors
[m1,n1]=hist(corr1,-3:0.1:3);
[m2,n2]=hist(corr0,-3:0.1:3);
[m3,n3]=hist(corrN,-3:0.1:3);
%Plot the histograms
plot(n2,m2/400*100,'red');
gtext('m=0');
hold;
plot(n3,m3/400*100,'k');
gtext('No watermark');
plot(n1,m1/400*100,'blue');
gtext('m=1');
xlabel('Detection value');
ylabel('Percentage of images');
```