



**HAL**  
open science

# Image Watermarking using Chaotic Watermark Scrambling and Perceptual Quality Evaluation

Rui Ye

► **To cite this version:**

Rui Ye. Image Watermarking using Chaotic Watermark Scrambling and Perceptual Quality Evaluation. 2013. hal-00861068

**HAL Id: hal-00861068**

**<https://hal.science/hal-00861068>**

Submitted on 11 Sep 2013

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# **Image Watermarking using Chaotic Watermark Scrambling and Perceptual Quality Evaluation**

Master: MDM

Laboratories: IRCCyN/IVC & IETR

Student: Rui YE

Supervisors: Florent Autrusseau

Safwan El Assad

Date: 22/08/2013



## **Abstract**

In this report, a watermarking method for grayscale images is proposed that is invisible and robust to certain attacks. Chaotic maps are used to generate the watermark which improves the security of the method. The crown watermark is embedded in the real part of DFT domain and the embedding position is determined by the SURF algorithm. The peak signal-to-noise ratio is used to evaluate the perceived quality of the marked image. Normalized cross correlation is used for watermark detection. The original image is not required during the detection. Experiments are conducted to evaluate the robustness of the proposed method against different attacks on several images.

# Contents

|   |           |
|---|-----------|
| <b>1. Introduction.....</b>                 | <b>1</b>  |
| 1.1 Digital Watermarking.....               | 1         |
| 1.2 Chaotic Maps.....                       | 7         |
| 1.3 The SURF Algorithm.....                 | 8         |
| 1.4 Related Works.....                      | 10        |
| <b>2. Proposed Watermarking Method.....</b> | <b>15</b> |
| 2.1 Watermark Generation.....               | 15        |
| 2.2 Watermark Location.....                 | 17        |
| 2.3 Watermark Embedding.....                | 19        |
| 2.4 Watermark Detection.....                | 26        |
| <b>3. Experimental Results.....</b>         | <b>34</b> |
| 3.1 Parameter Selection.....                | 34        |
| 3.2 Experiment Results.....                 | 39        |
| <b>4. Conclusion.....</b>                   | <b>42</b> |
| <b>References.....</b>                      | <b>43</b> |

# 1. Introduction

In this section, we will briefly introduce digital watermarking technique, along with some other techniques that are used in our watermarking method, such as chaotic maps and the SURF algorithm. Also, a review of previous watermarking methods that have been proposed are provided, which analyses the strengths and weaknesses of these methods.

## 1.1 Digital Watermarking

The rapid development of digital services and computer networks in modern days makes image distribution more convenient than ever. The digital representation of copyrighted images certainly offers many advantages, but also enables illegal use of digital images, which poses a serious threat to the rights of content owners, causing a pressing need for copyright protection and multimedia security. The conventional cryptographic system can prevent illegal access to the encrypted data without a key, but once the data is decrypted, there is no way to track its reproduction and redistribution. One solution to this problem is digital watermarking technique, which embeds information about the copyright owner permanently in the data and remains present within the data after any decryption process.

Digital watermarking is a process that imperceptibly alters a host media to embed a message within that host media. The embedded watermark is a signal that contains ownership information. It should be invisible to human eyes and can only be detected by a computer program.

Digital watermarking techniques have a wide variety of applications. It can be used for copyright protection, to identify the content owner, for fingerprinting, to identify the buyer of the content, for copy control, to prevent illegal copying, for broadcast monitoring, to determine royalty payment, and for content authentication, to determine whether the data has been altered in any manner from its original form. There are many other applications which all take advantage of watermarking for their own purposes.

There are a number of desirable properties that a watermark should possess. These include invisible to human vision, robust to common distortions of the signal, resistant to hostile attacks which aim to remove the watermark, sufficient embedding capacity, and a proper algorithm complexity. Notice that these requirements are application dependent, which means with different target applications, the requirements we should focus on varies as well. For some applications robustness may be the first priority, while for other applications the main concern is the algorithm complexity. Therefore, it is important to carefully choose the properties to work on based on the requirement of application. These properties are discussed in more detail next.

**Invisibility:** the watermark should not be noticeable and should not degrade the quality of the content. This property is often referred to as imperceptibility as well,

which means the presence of the watermark should not interfere with the perceived quality of the host media, a perceptual similarity between the watermarked and unwatermarked content must be preserved. To achieve this goal, some of the early watermarking methods often place the watermark in perceptually insignificant regions of the data. However, this may compromise other properties of the watermark such as robustness. As technology develops, many alternative methods were proposed to obtain an acceptable level of invisibility while maintaining robust against distortions. Depends on the application, sometimes a modest perceptible watermark can be accepted in exchange for a higher robustness. Since an observer is not likely to compare the watermarked content with the original one, a slight difference may not be noticed.

**Robustness:** a watermark should be able to survive common signal distortions such as spatial filtering, digital-to-analog and analog-to-digital conversion, lossy compression, printing and scanning and contrast enhancement, as well as common geometric distortions such as rotation, translation, scaling and cropping. A watermark is not required to be robust against all possible distortions, just the distortions that are likely to occur during the time between embedding and detection. In some applications like authentication, the robustness of a watermark is not even desirable at all, any distortion imposed on the image should cause the watermark to be lost. Meanwhile, there are other applications in which the watermark should be very robust due to the unpredictability of distortions that an image may undergo. Note that a robust watermarking method requires the watermark to not only remain present in the image after distortions, but also remain detectable in the detector. For example, watermarks embedded by some methods can remain in the image after distortion, but cannot be detected during detection. In this case, the watermarking method is considered not robust enough.

**Security:** a watermark should be able to resist intentional tampering designed to remove the watermark such as unauthorized removal, unauthorized embedding and unauthorized detection, as opposed to common signal distortions. One form of unauthorized removal attacks is collusion attack, this attack is possible when several copies of the same content are available and contain different watermarks, then an adversary can combine them to produce a copy without watermark. Security is more important to applications whose algorithm is known to the public or the original unmarked content is available, which allows anybody to decode the watermark. If the original content or the key to generate the watermark is only known to the owner and buyer, then the watermark is more tamper resistant and secure. In this case, if the attacker wants to make sure that the watermark is completely destroyed, the only way is to increase the strength of attacks, which may cause the image fidelity to be severely damaged and holds no value to the attacker.

**Capacity:** the maximum number of bits that can be hidden in a given data. Different applications may require different embedding capacity. For example, copy control applications only need 4-8 bits of information over a period of time while broadcast monitoring applications need 24 bits of information to identify all commercials. However, the more bits we embed within the data, the more visible the

watermark may be. Obviously, it is application dependent, and this is especially important to public applications.

**Complexity:** the computational costs of the encoder and decoder. In general, the cost should be limited in a certain range. However, this requirement may constrain the complexity of the watermark, causing the watermark to be simple and more vulnerable to intentional tampering. But notice that the computational ability of computers improves very fast, and some computationally unrealistic methods today may become possible in the future, which allows us to spend more costs to deal with issues such as geometric distortions and intentional tampering.

As mentioned before, these requirements of a watermark are application dependent and conflicting with each other. For example, when the robustness increases, the invisibility usually decreases, and similarly, when the capacity decreases, the invisibility typically increases. Therefore, it is important to find the best tradeoff between these requirements for a given application. In this report, we mainly focus on the invisibility and robustness tradeoff. The goal is to design a watermarking method which is invisible and able to withstand geometric distortions, known as the most devastating attacks on watermarking methods.

A watermarking scheme usually contains two parts, the embedding part to embed the watermark in a given image, and the detection part to determine if the watermark is detectable or not.

During the embedding process, the original unmarked image is first transformed into the domain we want to embed the watermark in, and then the watermark is generated using a secret key. After this, we embed the watermark within the chosen domain and perform an inverse transform to obtain the watermarked image. The block diagram of this process is shown below:

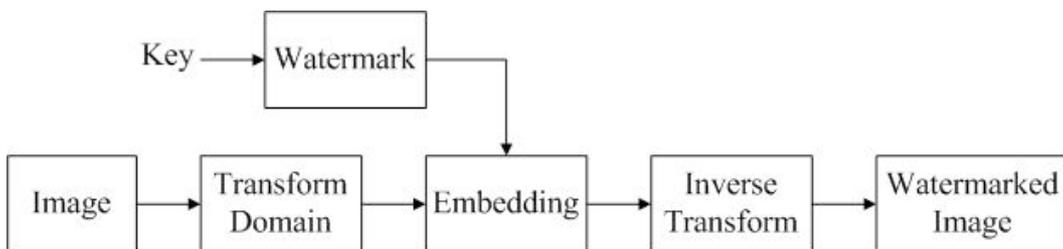


Figure 1. Block diagram of embedding process

The key is used in conjunction with watermarking algorithm to embed the watermark into a host media. The key may control the locations where the host media is modified (the embedding path) or may be used to generate other primitives that enter the embedding process. If an adversary do not have access to the key, then it is extremely difficult for them to remove the watermark without causing significant degradation in the fidelity of the image. Therefore, the watermarking algorithm can be made public without compromising the security of the watermark if the key remains secret.

When we embed the watermark  $X$  into the original image  $V$  to obtain the watermarked image  $V'$ , we set an embedding strength  $\alpha$  to determine the extent to

which the watermark alters the original content. Three most commonly used formulas to compute  $V'$  are [3]:

$$v'_i = v_i + \alpha \cdot x_i \quad (1)$$

$$v'_i = v_i \cdot (1 + \alpha \cdot x_i) \quad (2)$$

$$v'_i = v_i \cdot (e^{\alpha \cdot x_i}) \quad (3)$$

Formula (1) is called additive embedding, and it simply adds the values of coefficients in the original content with the values of the watermark. This may not be appropriate when the values of  $V$  vary widely. For example, if  $v_i = 10^5$  then adding 100 may not be sufficient to make the watermark detectable, but if  $v_i = 10$  then adding 100 may destroy the fidelity of the image. Formula (2) is called multiplicative embedding, and it is image dependent because how much we alter the original content depends on the value of  $V$ . This way it is more robust against such variation in scale. Formula (3) gives similar results as Formula (2) when  $\alpha x_i$  is small, and can be viewed as the logarithm version of Formula (1).

The choice of embedding domain is also very important. Earlier watermarking methods simply embed the watermark in the spatial domain. Certainly it is easy to achieve invisibility and a high capacity, but the robustness is very weak, which makes the watermark unable to resist any distortions, unintentional or intentional. Later, people start to implement the watermark in the frequency domain using some discrete transform to exploit the properties of these transform domain. Thus, after the implementation and transformation back to the spatial domain, the energy of the watermark is distributed over the whole image, which improves the invisibility and robustness at the same time.

The most commonly used frequency domains are discrete cosine transform (DCT), discrete wavelet transform (DWT), and discrete Fourier transform (DFT), there are also some methods that use Fourier-Mellin transform for its invariance to rotation and scaling. Each method has its own strengths and drawbacks. DCT-based approach has a strong robustness against JPEG compressions since JPEG compressions itself takes place in DCT domain. In addition, it is robust to HVS based quantization. However, it is not robust to geometric distortions. DWT-based approach is robust to JPEG 2K compression, low-pass and median filtering, but not robust to geometric distortions and cannot be adapted to human visual system (HVS). DFT-based approach is very robust against geometric transformations since it is translation invariant and rotation resistant. It is also suitable for HVS models. However, it introduces round-off errors, which may cause loss of quality and errors in watermark extraction. Fourier-Mellin approach is rotation and translation invariant, which makes it very robust against geometric distortions. But it has a weak resistance to lossy compression and a high computational complexity. The advantages and disadvantages of these different embedding domains are listed in the table below:

| Embedding Domains | Strengths  | Drawbacks   |
|-------------------|--|---|
| Spatial           | <ul style="list-style-type: none"> <li>- Easily invisible</li> <li>- High capacity</li> </ul>  | <ul style="list-style-type: none"> <li>- Weak robustness</li> </ul>   |
| DCT               | <ul style="list-style-type: none"> <li>- Robust to JPEG compression</li> <li>- Robust to HVS based quantization</li> </ul>           | <ul style="list-style-type: none"> <li>- Not robust to geometric distortions</li> </ul>   |
| DWT               | <ul style="list-style-type: none"> <li>- Robust to JPEG 2K compression</li> <li>- Robust to low-pass and median filtering</li> </ul> | <ul style="list-style-type: none"> <li>- Not robust to geometric distortions</li> <li>- Not adapted to HVS</li> </ul>             |
| DFT               | <ul style="list-style-type: none"> <li>- Robust to geometric distortions</li> <li>- Suitable for HVS models</li> </ul>               | <ul style="list-style-type: none"> <li>- Loss of quality and errors in watermark extraction</li> </ul>                            |
| Fourier-Mellin    | <ul style="list-style-type: none"> <li>- Rotation and translation invariant</li> <li>- Robust to geometric distortions</li> </ul>    | <ul style="list-style-type: none"> <li>- Weak resistance to lossy compression</li> <li>- High computational complexity</li> </ul> |

Table 1. Strengths and drawbacks of different embedding domains

Since the main goal of our work is to design a robust watermark, especially to geometric distortions, we choose to embed the watermark in DFT domain.

After the watermarks are embedded, the quality of a watermarked image needs to be evaluated. The most commonly used metrics for quality assessment in image processing are peak signal-to-noise ratio (PSNR), visual information fidelity (VIF), visual signal-to-noise ratio (VSNR) and the structural similarity (SSIM). However, the watermarking community mainly uses the PSNR or SSIM. Here we use PSNR. Admittedly, it does not perform very well in some circumstances, but still considered to be a fair indicator to provide qualitative rank order scores in overall. The value of PSNR is often given in decibels. Generally, values above 40 dB suggest low degradation, while values below 30 dB suggest low quality. So, if the PSNR of a watermarked image is above 40 dB, we consider the watermark to be quite invisible.

After the embedding, a detection process has to be proceeded to determine whether the embedded watermark is still present in the image after attacks. During detection, the watermarked and possibly attacked image is first transformed into the embedding domain where the watermark is embedded. Then we extract the watermark and compare it with the original watermark to see whether the watermark is able to survive from the attacks or not. The block diagram of this process is shown below:

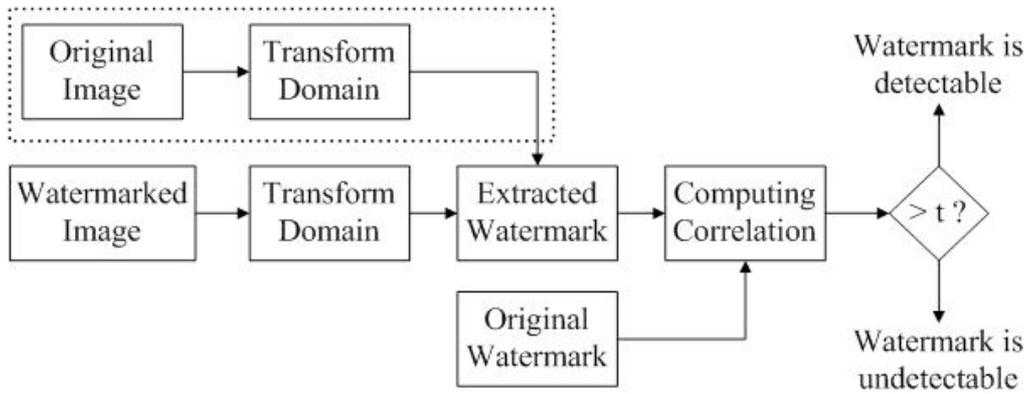


Figure 2. Block diagram of detection process

The dashed box in the block diagram marked the difference between two kinds of detectors, the blind detector and the non-blind detector. In some applications, the original unwatermarked image is available during detection. This will improve the detector performance significantly since the original image can be used to counteract any temporal or geometric distortions, which will reduce the damages caused by attacks to minimum. In other applications, the original image may be unavailable, so the detection must be conducted without the original.

The non-blind detector is defined as a detector that requires access to the original unwatermarked image in the detection process, or some information about the original image rather than the entire image. As contrary to non-blind detector, the blind detector is defined as a detector that does not require any information related to the original unwatermarked image. Whether a watermarking system employs blind or non-blind detector can be crucial in determining whether it can be used for a given application. Non-blind detector can only be used in those applications where the original document is available. In general, the original documents are only available in private applications, and therefore non-blind detectors cannot be used in public applications.

In this report, we use blind detector instead of non-blind detector in order to improve the adaptability of the proposed watermarking method. In this way, the watermarking system is no longer constrained by the presence of the original image, which makes it easier to adapt the method to other applications.

There are two important probabilities that should be estimated during detection, the false positive probability and the false negative probability. False positive probability refers to the probability to detect a watermark in an unmarked image, also known as false alarm. False negative probability refers to the probability of not detecting the watermark in a marked image, also known as false rejection. We should try to keep both probabilities as low as possible, since a high false detection rate is unacceptable for most watermarking methods. The required false positive probability depends on the application. Some applications have strictly specified the allowed value, while some are comparably less strict.

In order to determine whether the watermark is still present after attacks, we need to measure the similarity between the extracted watermark and the original watermark,

since it is highly unlikely that the two marks will be identical. There are many possible measures, and we choose to use normalized cross correlation for our method.

The normalized cross correlation is computed by first normalizing the two vectors to unit magnitude. This is typically done by subtracting the mean and dividing by the standard deviation. Then it computes the inner product between the two vectors to get correlation coefficients. In our case, the two vectors are the extracted watermark and the original watermark, and the result is used to measure the similarity between them. Most methods will set a threshold  $T$ , and check whether the maximum value is above  $T$ . If it is, then the image is watermarked. If not, the image is not watermarked. For our method, we do not use a threshold, instead we conduct a test called Grubbs' test to detect outliers. If the number of outliers is more than 1, which means there is more than one peak in the correlation map, then we consider the image to be watermarked. This will be explained in detail later.

## 1.2 Chaotic Maps

One of the novelties of our method is the use of chaotic maps to generate the watermark, which will enhance the resistance of the method against watermark removal attacks and improve the level of security.

Here we use a discrete-time dynamical system:  $X(n) = F[X(n-1)]$ . It is capable of complex and unpredictable behavior. A chaotic dynamical system is:

1. Deterministic, not random and unpredictable. This means that the system has no random or noisy inputs. The irregular behavior arises from the system's nonlinearity.
2. Aperiodic long term behavior for continuous-time dynamical system. Means that there should be trajectories which do not settle down to fixed points, periodic orbits or quasi-periodic orbits as  $t \rightarrow \infty$ .
3. Periodic behavior for discrete-time dynamical system.
4. Sensitive to initial conditions and initial parameters (Secret Key). This suggests that nearby trajectories separate exponentially fast, which means the system has positive Lyapunov exponent.

There are many useful properties of chaotic sequences to ensure security. First, it is easy to generate, a simple discrete-time dynamical system is capable to generate a complex and random like behavior sequences. Second, a chaotic signal is deterministic, not random, which allows us to regenerate it. This is fundamentally different from random sequences, since a random process is non-deterministic, and two successive realizations of a random process will give different sequences, even if the initial state is the same. Also, it has a broadband spectrum. Third, a chaotic signal is extremely difficult to predict because of the high sensitivity of the secret key. A slightly difference of the secret key will result in an entirely different chaotic sequence, which will make it extremely difficult for attackers to regenerate or remove the watermark. Thus, the level of security will be significantly improved. Fourth, the number of orbits in finite region of phase space is very big, which makes it almost impossible to predict the sequences.

There are many chaotic generators which can be used to generate the chaotic sequences, such as Skew Tent map, Arnold cat map, PWLCM map, etc. In this work, the discrete Skew Tent map is used to generate the chaotic sequences. It is described by:

$$X(n) = F[X(n-1)] = \begin{cases} \left\lfloor 2^N \times \frac{X(n-1)}{P} \right\rfloor & \text{if } 0 < X(n-1) < P \\ 2^N - 1 & \text{if } X(n-1) = P \\ \left\lfloor 2^N \times \frac{2^N - X(n-1)}{2^N - P} \right\rfloor & \text{if } P < X(n-1) < 2^N \end{cases} \quad (4)$$

where  $P$  is the control parameter ranging from 1 to  $2^N-1$ , and  $N$  is the finite precision equal to 32 bits.

However, a finite precision will lead the chaotic sequences into periodic orbits, while solving the periodic effect of finite precision produce the balance problem of the sequences. Perturbation technique is used in order to avoid this effect of the finite precision. Perturbation has some desirable characteristics such as controllable long cycle length and uniform distribution, while not degrading the good statistical properties of chaos dynamics. By adding a linear feedback shift register (LFSR), perturbation is done every delta number of iterations, where delta is orbit of the chaotic map without perturbation. The block diagram of this perturbation process is shown below:

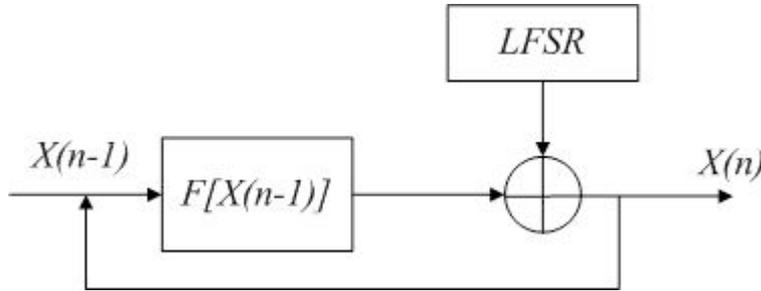


Figure 3. Block diagram of perturbation process

After we generate the chaotic sequence using the Skew Tent map, we get a sequence with integer values ranging from 1 to  $2^N-1$ . Because of its noise-like appearance, we can easily normalize it into the range we want and use it to form the watermark. In our case, it will be normalized into decimal values ranging from -1 to 1. Also, it has very good auto and cross correlation properties, which in theory may help improve the performance of the detector. One thing we know for sure is that the use of chaotic sequences will improve the level of security due to the secrecy of the key.

### 1.3 The SURF Algorithm

Another novelty of our method is the use of the SURF algorithm to find feature points in the Fourier spectrum as embedding positions. And during the detection, we

apply SURF again on the watermarked image to try to find the locations where the watermarks are embedded. This will make the watermarking method image dependent, since SURF will find different feature points in different images, makes it harder for attackers to predict the watermarking location and easier for us to extract the watermark.

SURF is short for Speed-Up Robust Features. It is a scale and rotation invariant interest point detector and descriptor. SURF approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster. This is achieved by relying on integral images for image convolutions; by building on the strengths of the leading existing detectors and descriptors (specifically, using a Hessian matrix-based measure for the detector, and a distribution-based descriptor); and by simplifying these methods to the essential. This leads to a combination of novel detection, description, and matching steps.

SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images. It uses an integer approximation to the determinant of Hessian blob detector, which can be computed extremely quickly with an integral image (3 integer operations). For features, it uses the sum of the Haar wavelet response around the point of interest. Again, these can be computed with the aid of the integral image.

An integral image (also known as summed area table) is a data structure and algorithm for quickly and efficiently generating the sum of values in a rectangular subset of a grid. It allows for fast computation of box type convolution filters. The entry of an integral image  $I_{\Sigma}(\mathbf{x})$  at a location  $\mathbf{x} = (x, y)^T$  represents the sum of all pixels in the input image  $I$  within a rectangular region formed by the origin and  $\mathbf{x}$ . The value at any point  $(x, y)$  in the integral image is just the sum of all the pixels above and to the left of  $(x, y)$ :

$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y) \quad (5)$$

The Hessian matrix is a square matrix of second-order partial derivatives of a function. It describes the local curvature of a function of many variables. SURF bases the detector on the Hessian matrix because of its good performance in accuracy. More precisely, it detects blob-like structures at locations where the determinant is maximum. Given a point  $\mathbf{x} = (x, y)$  in an image  $I$ , the Hessian matrix  $H(\mathbf{x}, \sigma)$  in  $\mathbf{x}$  at scale  $\sigma$  is defined as follows:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (6)$$

where  $L_{xx}(\mathbf{x}, \sigma)$  is the convolution of the Gaussian second order derivative  $\frac{\partial^2}{\partial x^2} g(\sigma)$  with the image  $I$  in point  $\mathbf{x}$ , and similarly for  $L_{xy}(\mathbf{x}, \sigma)$  and  $L_{yy}(\mathbf{x}, \sigma)$ .

Generally speaking, the task of finding image point correspondences can be divided into three main steps. First, interest points are selected at distinctive locations

in the image, such as corners, blobs, and T-junctions. The most valuable property of an interest point detector is its repeatability. The repeatability expresses the reliability of a detector for finding the same physical interest points under different viewing conditions. This property is extremely important to our watermarking method because one of the main challenges in the detection process is to find the same interest points even when the image has gone through signal or geometric distortions. Next, the neighborhood of every interest point is represented by a feature vector. This descriptor has to be distinctive and at the same time robust to noise, detection displacements and geometric deformations, which is very important to the robustness of the watermark. Finally, the descriptor vectors are matched between different images. The matching is based on a distance between the vectors.

As for SURF, it is partly inspired by the SIFT descriptor. Similarly to the SIFT approach, SURF selects interest points of an image from the salient features of its linear box-space, a series of images obtained by the convolution of the initial image with box filters at several scales. Then SURF builds local features based on the histograms of gradient-like local operators. However, the standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.

In conclusion, SURF is a fast and robust algorithm for local feature point detection and comparison. It has a comparable and sometimes even better performance than the state-of-the-art detectors, and a high repeatability which is advantageous for watermarking methods to find embedding positions. And the most important is the speed of the detector, which outperformed most of the existing methods. But notice that SURF is usually used in spatial domain, while in our method it is used in a much noisier environment of DFT domain. Therefore, the actual performance of SURF in Fourier spectrum has to be tested to prove its adaptability in a highly noisy environment.

## 1.4 Related Works

Several previous digital watermarking methods have been proposed. This is not a complete list of all the watermarking methods, just the ones that are closely connected to our method.

Early works of watermarking mainly focused on hiding information within a signal without considering other requirements such as robustness and security. Tamper resistance is not an issue if the communication channel is covert and only the communication parties are aware of it. Therefore, early works of watermarking can be seen as steganography.

Solachidis *et al* <sup>[1]</sup> proposed a blind image watermarking method that is resistant to geometric transformations. The watermark is embedded on a ring in the DFT domain using a private key, since embedding in the Fourier domain has certain advantages for scaling and rotation invariance. Also it is embedded in the middle frequency of the magnitude of Fourier spectrum to achieve balance between robustness and invisibility. The watermark possesses circular symmetry to solve

rotation invariance in an easy way. Correlation is used for watermark detection and the original image is not required in detection. This method is robust against several image processing attacks, especially geometrical distortions. And if we want to search only for scaling or for cropping or for some rotation angles only, the calculation is very fast. However, the algorithm does not perform well after a rather large rotation, and it uses additive embedding not multiplicative embedding, so it is not image dependent.

Solachidis *et al* <sup>[2]</sup> also presented a statistical analysis of the behavior of a blind robust watermarking system based on pseudorandom signals embedded in the magnitude of the Fourier transform domain, and the design of an optimal detector for multiplicative watermarking embedding. The watermark is embedded in the DFT domain so that it can be more robust to geometric distortions. The DFT magnitude distribution is analytically calculated which is proven to be different than the generally used Weibull distribution. It also constructs the optimal detector according to the Neyman-Pearson criterion instead of using correlation detector since correlation detector is optimal only in case of the signal follows a Gaussian distribution. However, it did not give a comparison between the proposed detector and the Weibull one. And in threshold estimation step, they have to know the detector distribution in case of erroneous watermark detection and they assume that the distribution is Gaussian, which may not be the case under some circumstances.

Cox *et al* <sup>[3]</sup> proposed a robust watermarking scheme which takes advantage of spread spectrum. It spread the watermark over many frequency bins so that the energy in any bin is very small and undetectable, thus ensuring imperceptibility. Also, it is argued that the watermark should be inserted in the perceptually most significant components of the data and composed of random numbers drawn from a Gaussian distribution since watermarks in perceptually insignificant regions can be easily removed, this has a huge influence on several watermarking methods proposed later. The strength of this method is the embedding of watermark in perceptually most significant regions, which makes the watermark robust against common signal processing, and can withstand collusion attacks. The weakness is that the extraction progress needs the original unmarked document, thus only allows the watermark to be extracted by the content owner. Moreover, it is not suitable against geometric distortions.

Cox *et al* <sup>[4]</sup> also provided a review of the properties and a basic framework of watermarks, along with some proposed watermarking methods of different kinds. They focused on the desirable characteristics of a watermark for copyright control, including invisibility, robust to common signal distortions, resistant to tampering, bit rate, modification and multiple watermarks, and scalability. And they suggested that a watermark should be placed in the perceptually significant regions of the data to enhance robustness. Also, the PSNR should be high for the watermark to be unnoticeable. Followed by that, they introduced a mathematical framework to analyze the existing watermarking methods, and reached a conclusion that a prefiltering and nonlinear insertion is to be preferred. Finally they outlined many proposed watermarking scheme and analyzes the strengths and weaknesses regarding to

prefiltering and image spectral shaping.

O'Ruanaidh *et al* <sup>[5]</sup> described how Fourier-Mellin transform-based invariants can be used for digital image watermarking. The embedded watermarks are designed to be robust against any combination of translation, rotation and scale. The watermark is also embedded in the perceptually significant components of the image and it also uses spread spectrum technique to achieve security and error reduction. The importance of invertibility of the integral transform invariants was emphasized. One of the significant points of this method is the novel application of the Fourier-Mellin transform to digital image watermarking. There are several advantages in using integral transform domain marks. First, the transform space contains a large number of samples which can be used to hide a spread spectrum signal. In addition, it makes the method robust to changes in scale and rotation. However, it has a weak resistance to lossy image compression and cropping, and cannot resist changes in aspect ratio or shear transformations. Also, in practice the inverse log-polar mapping is a computational bottleneck.

Poljicak *et al* <sup>[6]</sup> proposed a blind watermarking method which minimizes the impact of the watermark implementation on the overall quality of an image. The watermark is embedded in magnitudes of the Fourier transform, and the obtained results were used to develop a watermarking strategy that chooses the optimal radius of the implementation which maximizes the PSNR value to minimize quality degradation. It is image dependent because with different images, the radius of the watermark changes accordingly to provide a better result. It has a high level of security due to the varying radius of implementation. It is computationally simple, device independent, adaptable and more robust while maintaining the same influence on overall quality of a watermarked image. However, the watermark is composed of circular dots of random noise, resulting in a low capacity and may not be suitable for other shapes of watermarks. And the PSNR may not be the best metric to evaluate the quality of images if it wants to obtain the optimal radius of implementation.

Tataru *et al* <sup>[7]</sup> proposed a modified DCT watermarking method for grey images realized in frequency domain, with resilience to certain watermarking attacks and an improvement in terms of security, assured by the use of chaotic sequences. It also designed a watermark removal attack based on neighbor's mean to evaluate the performance between the proposed chaotic method and the standard method proposed in Zhao *et al* <sup>[12]</sup>. The main novelty of this method is the use of chaotic maps to scramble the watermark. It eliminates the inconvenience of the original standard method brought by the raster scan of the blocks and the fixed position used at the insertion. It is proven to be imperceptible, robust and most of all much more secure than the standard method. However, it uses a length of the watermark smaller than the capacity of the host image for security reasons, which may restrict the use of this method if the application requires a large capacity. Also, the PSNR decreases slightly with the chaotic method.

Solanki *et al* <sup>[8]</sup> proposed a watermarking method to hide information into images that achieves robustness against printing and scanning with blind decoding, in which the original image is not required at the decoder to recover the embedding data. A

significant contribution of this paper is that it proposed a technique to estimate and undo rotation. The method is based on the fact that laser printers use an ordered digital half toning algorithm for printing, and embedding in the transform domain with synchronization and error correction using powerful turbo-like channel codes. The proposed method named “Selective Embedding in Low Frequencies (SELF)” is based on the DFT magnitude, in which information is hidden only in dynamically selected low frequency DFT coefficients. The strength of this method is that there is no penalty in hiding rate for achieving robustness against rotation. Moreover, the estimation and automatic derotation allows more information to be hidden because of its accurate estimation of the rotation angle. However, it cannot be applied to a general rotation attack, and only has a good performance for watermarking applications if the watermark sequence is known to the decoder and can be correlated with the hidden data to detect the watermark.

Solanki *et al* <sup>[9]</sup> introduced an improved watermarking system of their last method. They propose two methods to hide information into images that achieves robustness against printing and scanning with blind decoding, namely “Selective Embedding in Low Frequencies (SELF)” (as mentioned in the last method) which hides information in the magnitude of selected low-frequency DFT coefficients, and “Differential Quantization Index Modulation (DQIM)” which embeds information in the phase spectrum of images by quantizing the difference in phase of adjacent frequency locations. A significant contribution of this method is a systematic analytical and experimental modeling of the print-scan process. It reaches the conclusions that during print-scan process, the low and mid frequency coefficients are preserved better than high ones, high magnitude coefficients are preserved better than low ones, and the difference in phase of adjacent frequency locations is preserved for the high magnitude coefficients. Several hundred information bits can be embedded into images with perfect recovery against the print-scan operation compared to other methods which embeds only a single bit or a few bits of information. The hidden images also survive several other attacks, such as Gaussian or median filtering, scaling or aspect ratio change, heavy JPEG compression, and rows and/or columns removal. However, the volume of embedding of DQIM is less than SELF since it is hard to embed data in the phase spectrum without introducing much perceptual distortion. And it is only suited for rotation attacks whose printing process is based on ordered digital halftoning algorithm.

Voloshynovskiy *et al* <sup>[10]</sup> proposed a new content adaptive stochastic approach which is based on the computation of a Noise Visibility Function (NVF) that embeds the watermark into the cover image according to local image properties, identifying textured and edge regions where the mark should be more strongly embedded. It examines two NVFs based on non-stationary Gaussian model and stationary Generalized Gaussian model respectively, and shows that the problem of the watermark estimation is equivalent to image denoising and derive content adaptive criteria. It is applicable for very different types of images, and not constrained with the identification of an adequate set of parameters to be determined before the identification of the local characteristics. Also, it may be applied to different domains.

Meanwhile, some changes can be made to further improve this method. For example, in content adaptive watermark embedding, the watermark strength in very flat regions should be replaced with an image-dependent variable instead of a fixed value, which can take the luminance sensitivity of HVS into account.

LeCallet *et al* <sup>[11]</sup> presented a review of both subjective and objective quality assessment of images and video in the field of watermarking and data hiding applications, in which the first aim is to highlight the deficiencies of some metrics for data hiding purposes. A deep review of subjective experiment protocols has been carried out and both usual statistical metrics, and a few advanced objective metrics, have been detailed. First it presented a quality benchmark, which is used to determine among several existing objective quality metrics, the one that would best predict the subjective scores. Then it argued that PSNR may not be the best metric to evaluate the quality of a watermarked image. Although one objective metric has provided a rather good prediction of the MOS, existing objective metrics may not yet be efficient enough to assess the quality of one single embedding technique for various contents or for different embedding strengths. For future work, further research could be devoted to the development of an efficient quality metric within the aforementioned range, and the design of a metric giving the right weight to perceptual ineffective geometric distortions is still a challenging matter of research.

## 2. Proposed Watermarking Method

In this section, we will discuss our proposed watermarking method in detail, including how to generate the watermark, how to choose the embedding location, and the detailed process of watermark embedding and detection.

### 2.1 Watermark Generation

There are many kinds of watermarks proposed in the past. Some spread the watermark throughout the spectrum of an image, over many frequency bins so that the energy in any one bin is very small and undetectable; some embed dots circularly around the center of the spectrum. While in our method, we will embed a crown-shaped watermark composed of decimal values ranging from -1 to 1.

The reason the watermark is crown-shaped is that we do not want the embedding watermarks interfere with the interest points found by SURF. If the watermark is a filled circle, then the property of the interest point may be changed by the watermark, causing SURF to lose this interest point during detection. Therefore, the inner radius of the crown watermark should be large enough to minimize its impact on interest points to minimum. Moreover, the watermarks are composed of sequences of decimal values instead of binary values because binary watermarks are less resistant to tampering and more visible.

As mentioned before, for our method, we will take advantage of chaotic maps, to be specific, the discrete Skew Tent map, to generate chaotic sequences and use them to construct the watermark. The sequences are generated using Formula (4), and the values are between 1 and  $2^N-1$ , where  $N$  is the finite precision equal to 32 bits. The key used to generate the sequences is comprised of 4 parameters: the control parameter  $P$ , the initial value of the sequence  $X_1$ , the initial register value of the perturbation  $Q_0$ , and the orbit of the chaotic map without perturbation  $\Delta$ . The first three parameters are generated randomly, while the last parameter  $\Delta$  is set to be 31, which means perturbation is done every 31 number of iterations.

In fact, the setting of parameter  $\Delta$  is very important to our method. Because when  $\Delta$  is set to a larger value, the orbit of the chaotic sequence appears to be predictable, which means despite the uniform distribution of the sequence, the organization of it is not random. This will result in the structure of the watermark appears to be regular, which will affect the invisibility of the watermark and the performance of the detector. Therefore in our method, we set  $\Delta$  to be a small value as 31.

The key of the sequence is only known to the content owner, makes it almost impossible for attacks to regenerate the watermark, which improves the level of security significantly. The polynomial equation used for the perturbation is:

$$g(x) = x^{15} + x^{13} + x^{10} + x^1 + 1 \quad (7)$$

After the chaotic sequence is generated, it has to be normalized into values of -1 to 1 to form the watermark. In order to achieve this, we use the following formula:

$$X = \begin{cases} \frac{X^\mu}{2^{N-1}} & \text{if } 1 \leq X^\mu \leq 2^{N-1} - 1 \\ \frac{X^\mu - 2^N}{2^{N-1}} & \text{if } 2^{N-1} \leq X^\mu \leq 2^N - 1 \end{cases} \quad (8)$$

where  $X^\mu$  is the value in the chaotic sequence before normalization, and  $X$  is the value after normalization.

After the normalization, we get a sequence with decimal values ranging from -1 to 1, with a uniform distribution and a mean value close to 0. The length of the sequence depends on the size of the watermark. It should be greater than the area of the circumscribed square of the watermark, since the values in the sequence will be filtered by a ring and used to construct the watermark. Also, the first 100 values are discarded since they have not reached to a stable periodic orbit.

With the normalized chaotic sequence, we can begin to form the watermark. First we load the sequence and save it as a square image, the size of which equals to the diameter of the watermark. Then we create an image of the same size, with a ring filled with values of 1, and the background is set to be 0. The radius and thickness of the ring equals to those of the watermark. By multiplying these two images, we can obtain the watermark with a zero background and a ring filled with decimal values of -1 to 1, generated by Skew Tent map. The watermark is saved for further use. This process is illustrated below:



Figure 4. The process of watermark generation

In order to evaluate the performance of the chaotic sequence generator, we designed a random noise generator for comparison purpose. The procedure of generating the watermark is similar to the chaotic generator, but instead of using chaotic map to generate the chaotic sequence, it uses a random generator to produce a sequence of random noise.

First we use the function `cvRNG()` in OpenCV to initialize the generator. It has one parameter called *seed*, which is a 64-bit value used to initialize a random sequence. This function initializes a random number generator state and returns the state. The pointer to the state can be then passed to the `cvRandReal()` function. `cvRandReal()` has one parameter *rng*, which is a random number generator state initialized by `cvRNG()`. The function returns a uniformly-distributed random floating-point number between 0 and 1. The sequence is then multiplied by 2 and minus 1, which produces a sequence with real values ranging from -1 to 1. This sequence is subsequently used to generate a watermark with random noise, and the generation process is the same as the chaotic watermark. Both the chaotic watermark

and the random-noise watermark will be tested to evaluate their performance when subjected to different kinds of attacks.

## 2.2 Watermark Location

After the watermark is generated, the next thing to determine is the embedding location of the watermark. As mentioned before, we decide to embed the watermark in the Fourier domain due to its robustness against geometric distortions. Now the question becomes, which locations in the Fourier domain do we want to embed the watermark on. In our method, we use SURF to help find these locations.

SURF is generally used to find correspondences between two images of the same scene or object. It detects interest points in a given image and matches them between different images. In our case, we use it to find interest points in DFT domain. While it is not originally designed for a highly noisy environment like DFT domain, the experiments we conduct show that SURF has a good adaptability and perform quite well in DFT domain. Moreover, it is very robust to different image transformations, which allows us to relocate the interest points we find in the original image after the image is watermarked. Even when the watermarked image has been attacked, it can still find some interest points near the original points, if not the exact ones. This is very important to our method since the performance of the detector highly depends on whether SURF can retrieve the interest points where the watermarks are embedded. After relocating the interest points, we can easily retrieve the watermarks and compare it with the original watermark.

Therefore, for a given image, we first compute its Fourier transform, split it into real and imaginary parts, and then compute the magnitude of the spectrum. After the magnitude coefficients of the spectrum is obtained, we then apply SURF on it to find interest points over the spectrum, as shown in Figure 5, where each white point represents a interest point found by SURF.

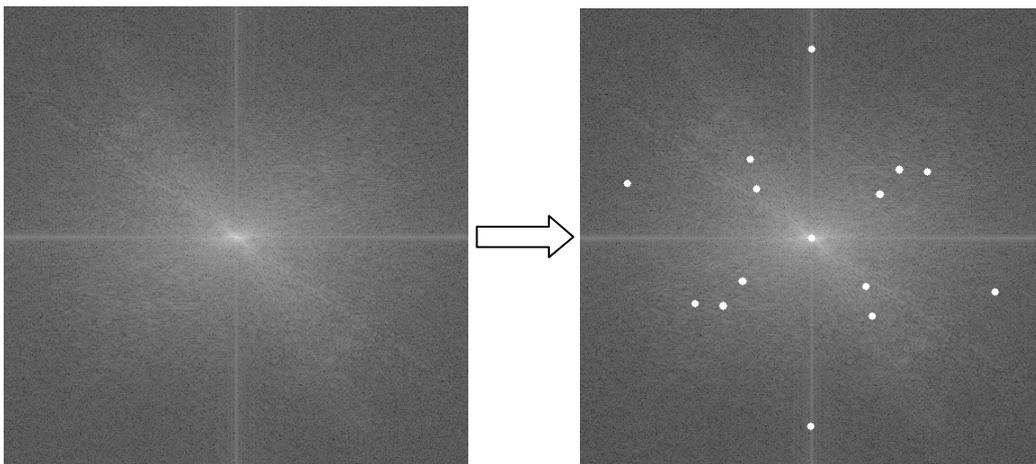


Figure 5. Interest points found by SURF in Fourier domain

However, not all interest points are suitable for embedding, we have to sift through these points and get rid of the ones that are not qualified as embedding

locations.

First, the watermarks should be embedded in the middle frequencies of the Fourier domain. Modifications in the low frequencies of the Fourier transform will cause visible changes in the spatial domain, which will degrade the perceived quality of the image severely. Embedding in the high frequencies of the Fourier transform is not robust enough against attacks since common distortions tend to affect the high frequencies more than lower frequencies. Thus, the watermarks should be added in the middle frequency range because, if carefully designed, it will be robust against distortions and invisible at the same time. Therefore, we need to filter out the interest points in the low and high frequencies, and only keep the ones in the middle frequencies, as shown in Figure 6.

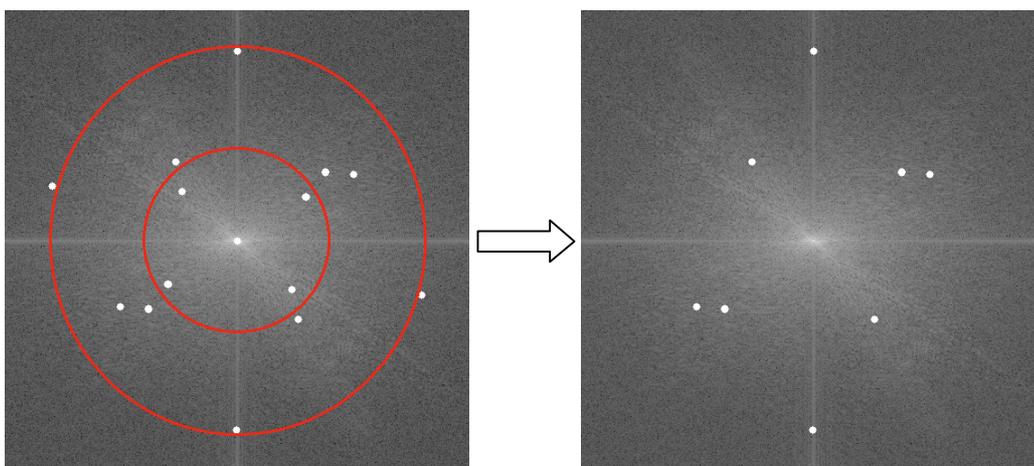


Figure 6. Remove the points that are in the low and high frequencies of the Fourier domain

We should also remove the points that are too close to others, because if not, the watermarks embedded around these points will overlap and affect the performance of the detector. Since SURF ranks the interest points by level of importance, if two points are too close, we will remove the less important one (also likely less robust). The distance between two points should be greater than the diameter of the watermark, thus ensures every watermark we embed will not overlap with others. This elimination process is shown in Figure 7.

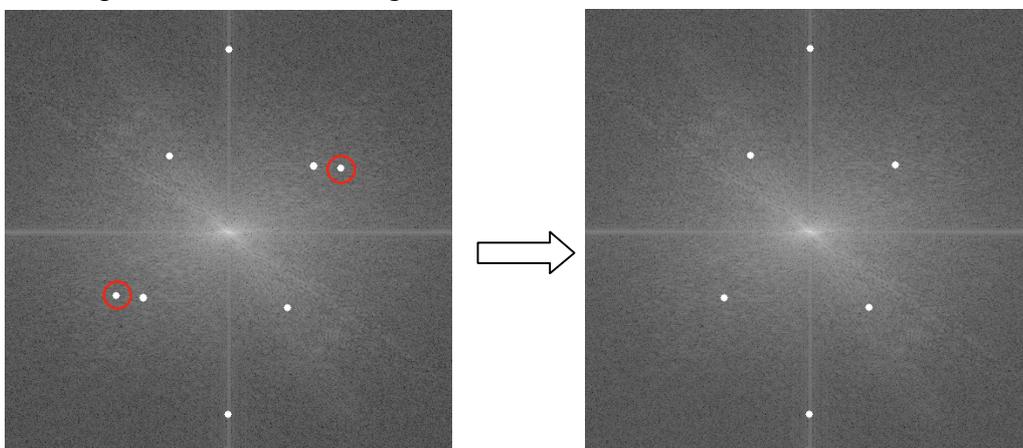


Figure 7. Remove the points that are too close to others and less important

After all the disqualified interest points are eliminated, the points left are the locations where we plan to embed the watermarks. The watermark will be embedded around every interest point, and the coordinate of the point is the center of the watermark.

## 2.3 Watermark Embedding

After the problems of how to generate the watermark and where to embed the watermark are solved, we can begin the embedding process.

The general scheme of our embedding process can be summarized as follows. For a given image, first perform Discrete Fourier Transform (DFT) on it, split the real part and the imaginary part of the spectrum and compute the magnitude coefficients of the Fourier spectrum. Then apply SURF algorithm on the magnitude coefficients to find some interest points. After filtering the interest points, only those points in the middle frequencies of the spectrum are kept as embedding locations. The crown watermark is generated using chaotic map and embedded around these interest points in the real part of the Fourier spectrum. Afterwards, the real part and the imaginary part of the Fourier spectrum are merged together and an inverse Fourier transform is performed to obtain the watermarked image. In addition, the PSNR of the watermarked image should be computed to evaluate its perceived quality. Both the watermark and the watermarked image are saved for further use during detection. This process will be analyzed in detail later. The block diagram of this process is shown below:

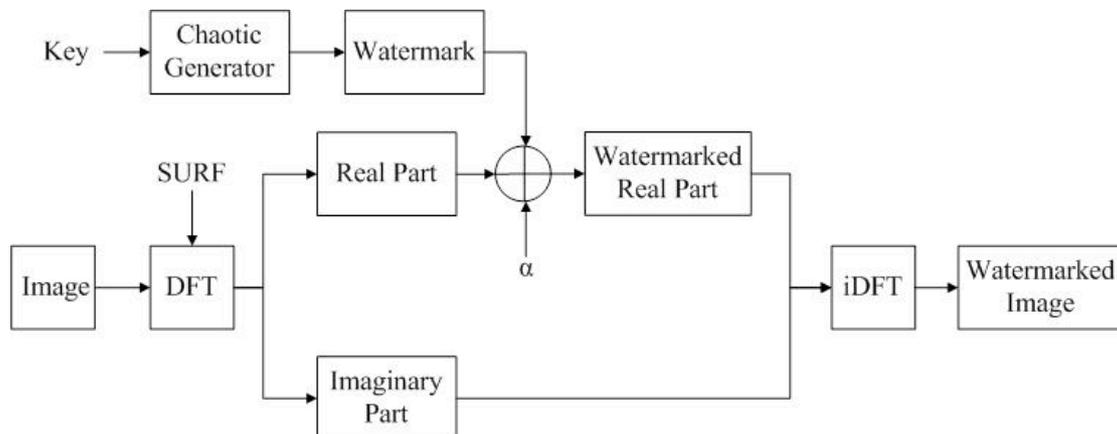


Figure 8. Block diagram of our embedding process

During embedding, the original image is first loaded and converted to grayscale. Then we perform DFT on the grayscale image. The Fourier Transform will decompose an image into its sinus and cosines components. In other words, it will transform an image from its spatial domain to its frequency domain. The idea is that any function may be approximated exactly with the sum of infinite sinus and cosines functions. Mathematically a two dimensional images Fourier transform is:

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) e^{-i2\pi(\frac{k_i}{N} + \frac{l_j}{N})} \quad (9)$$

$$e^{ix} = \cos x + i \sin x \quad (10)$$

Here  $f$  is the image value in its spatial domain and  $F$  in its frequency domain. The result of the transformation is complex numbers. Displaying this is possible either via a magnitude image and a phase image or via a real image and an imaginary image. Basically the magnitude of the Fourier transform at a point is how much frequency content there is, while the location is given by phase, which is the argument of the Fourier transform at a point. The real part is how much of a cosine of that frequency you need, while the imaginary part is how much of a sine of that frequency you need. However, throughout the image processing algorithms only the magnitude image is interesting as this contains all the information we need about the images geometric structure. This is also the reason why we apply SURF on the magnitude of the Fourier spectrum. Nevertheless, we intend to make some modifications of the image in these forms, and to retransform it we need to preserve all of these.

Since the image is converted to grayscale values between 0 and 255, the Fourier Transform also needs to be of a discrete type, resulting in a Discrete Fourier Transform. It is used to determine the structure of an image from a geometrical point of view. Note that changes in the spatial domain will affect the frequency domain as well. Circular shifts in the spatial domain do not affect the magnitude of Fourier transform, but will cause a linear shift in the phase component. Scaling in the spatial domain causes inverse scaling in the frequency domain. And rotation in the spatial domain causes the same rotation in the frequency domain.

In order to perform a DFT, first we need to expand the image to an optimal size. The performance of a DFT is dependent of the image size. It tends to be the fastest for image sizes that are multiple of the numbers two, three and five. Therefore, to achieve maximal performance it is generally a good idea to pad border values to the image to get a size with such traits.

Then we compute the discrete Fourier transform, and rearrange the quadrants of Fourier spectrum so that the origin is at the image center. The result of a Fourier Transform is complex, which implies that for each image value the result is two image values (one per component), the real part ( $Re$ ) and the imaginary part ( $Im$ ). Moreover, the frequency domains range is much larger than its spatial counterpart. Therefore, we store these results in a float format.

After we split the real part and the imaginary part of the Fourier spectrum, we can compute the magnitude of the Fourier spectrum using:

$$Mag = \sqrt{Re^2 + Im^2} \quad (11)$$

However, the dynamic range of the Fourier coefficients is too large to be displayed on the screen. There are some small and high changing values that we can't observe with human eyes. Therefore the high values will all turn out as white points, while the small ones as black. To solve this problem, we transform the linear scale to

a logarithmic one for visualization:

$$Mag' = \log(1 + Mag) \quad (12)$$

Finally we have to scale the image and throw away the new values introduced when we expand the image.

Now that we have obtained the magnitude of the Fourier spectrum, we can use SURF on it to find the interest points. The process of finding and filtering the interest points was described elaborately in section 2.2. Since the magnitude coefficients contain all the information about the geometric structure of the image, SURF is most likely to find the most robust interest points in it. The coordinates of these interest points are saved into a text file as future watermark embedding positions.

Once the embedding locations are determined, we can start to generate the watermark. The process of generating the watermark was explained in detail in section 2.1, in which we have described two ways to generate the watermark: using the chaotic sequence generator or random noise generator. The chaotic sequence generator is the main research object in this report, while the random noise generator serves as a comparison object to evaluate the performance of the former. The watermark is saved as a text image because we do not want any loss of precision of the values in the watermark. All the values are saved without any round off.

After all the preparatory work is completed, we can begin the actual embedding procedure to place the generated watermark on every embedding location in the Fourier domain. However, there is one thing that we should pay attention to, which is the symmetry property of the Fourier transform, specifically the fact that for time functions that are real-valued, the Fourier transform is conjugate symmetric. From this it follows that the real part and the magnitude of the Fourier transform of real-valued time function are even functions of frequency and that the imaginary part and phase are odd functions of frequency.

When we take the Fourier transform of a real function, for example a one-dimensional sound signal or a two-dimensional image, we obtain a complex Fourier transform. This Fourier transform has special symmetry properties that are essential when calculating and/or manipulating Fourier transforms.

Suppose in two dimensional we have an image  $f(x,y)$ , and the Fourier transform of this image can be written as:

$$F(u, v) = F_r(u, v) + iF_i(u, v) \quad (13)$$

The real part is symmetric and the imaginary part is anti-symmetric, where in two dimensions the symmetry conditions for the real part of the Fourier transform are given by:

$$\begin{aligned} F_r(u, v) &= F_r(-u, -v) \\ F_r(-u, v) &= F_r(u, -v) \end{aligned} \quad (14)$$

And the symmetry conditions for the imaginary part of the Fourier transform are given by:

$$\begin{aligned}
 F_i(u, v) &= -F_i(-u, -v) \\
 F_i(-u, v) &= -F_i(u, -v)
 \end{aligned}
 \tag{15}$$

Similarly the two dimensional power spectrum is also symmetric, with:

$$\begin{aligned}
 |F(u, v)|^2 &= |F(-u, -v)|^2 \\
 |F(-u, v)|^2 &= |F(u, -v)|^2
 \end{aligned}
 \tag{16}$$

This symmetry conditions are shown schematically in Figure 9, which shows a series of symmetric points.

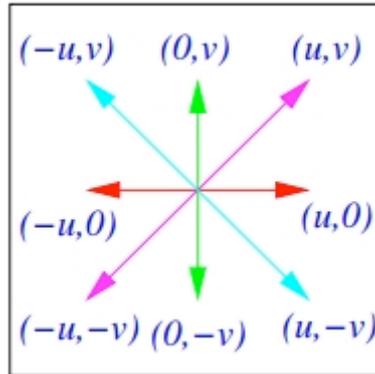


Figure 9. The symmetry conditions of Fourier transform

This symmetry property has a major significance in the digital calculations of Fourier transforms and should not be destructed. Since we are embedding the watermark in the real part of the Fourier transform, therefore the watermarks embedded in the first and second quadrants should be symmetric to the watermarks embedded in the third and fourth quadrants. Only in this way, the symmetry of the Fourier transform can be preserved.

Because of the symmetry of the Fourier transform, the interest points found by SURF are also central symmetric and the symmetry point is at the center of the image. For example, for image *Lena*, with given conditions, SURF can find four interest points in the middle frequencies of Fourier spectrum, which are central symmetric:

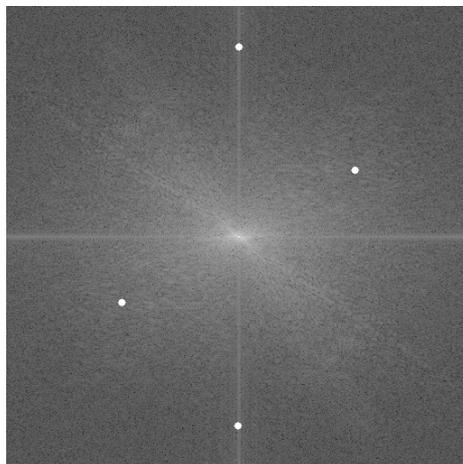


Figure 10. Interest points found by SURF for *Lena* image

First we need to create a new image with a size equal to the original image and set the background to 0. Then load the coordinates of the interest points found by SURF. We only need to go through half of the interest points which are in the third and fourth quadrants, and add the watermark around these points. After this, we create a central symmetric version of this image by performing a simultaneous horizontal and vertical flipping of the image. The last step is to simply add up these two images so that the watermarks in the first and second quadrants are symmetric to the watermarks in the third and fourth quadrants. The following image is marked as  $wm$ , which will be used in the next step:

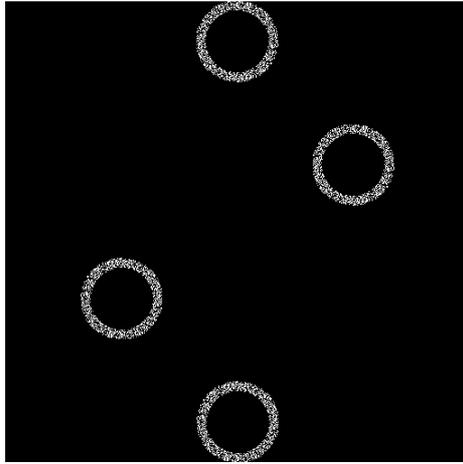


Figure 11. Final image  $wm$  with central symmetric watermarks

Finally, we embed these watermarks at the corresponding locations in the real part of the Fourier transform. As mentioned before, the most commonly used embedding functions are additive embedding and multiplicative embedding. In our method, we choose to use multiplicative embedding. It is more image-dependent because how much the watermark alters the original image depends on the value of the image at a given point. We also slightly modified the standard multiplicative embedding function (see Formula (2)) to make it more suitable for our embedding method. To embed a watermark  $X$  in the original image  $V$  with the embedding strength  $\alpha$  to obtain the watermarked image  $V'$ , the embedding function we use is as follows:

$$v_i' = \alpha \cdot x_i \cdot |v_i| \quad (17)$$

We load the coordinates of the interest points again and go through the real part of the Fourier transform, and every time we reach a watermarking location (the ring area around the interest point), the original value is replaced by the result of multiplying the absolute value of it by the value in  $wm$  at the same location and the embedding strength. The reason for multiplying the watermark by the absolute value of the real part is that we want the sign of the modified coefficients to be consistent with the sign of the watermark. If the value of the watermark is positive, then the value of the corresponding real part after embedding is also positive. Accordingly, if

the value of the watermark is negative, the value of the corresponding real part after embedding is negative as well. This way, for every modified coefficient, the sign of the coefficient after embedding would be the same with the sign of the watermark. Since we use cross correlation to evaluate the similarity between the watermark and the watermarked image during detection, keeping their signs consistent will certainly improve the correlation value, which will consequently improve the performance of the detector.

After the embedding is finished, the real part and the imaginary part are merged together to form the new Fourier domain containing the watermarks. The Fourier domain after embedding is shown below (contrast of the image is enhanced to make the watermark more visible):

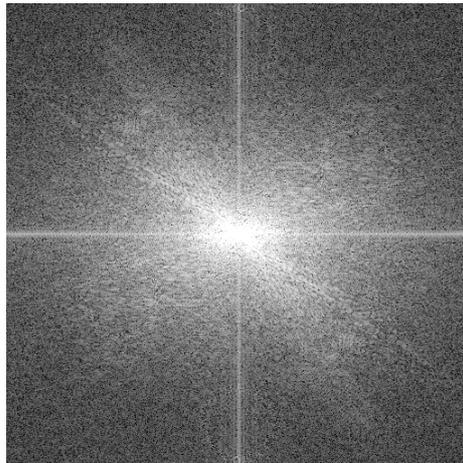


Figure 12. Fourier domain after embedding

The reason we are embedding in the real part of the Fourier transform instead of the magnitude is that the real part contains both positive and negative values, while the magnitude is consisted of all positive values. The embedded watermarks have both positive and negative values too like the real part, so the correlation between the watermark and the real part is likely to be higher. If the embedding takes place in the magnitude, then some of the positive values may be changed into negative values. In fact, the embedding function we use is specifically designed for embedding in the real part of the Fourier transform.

Also, we have embedded several watermarks in various locations instead of one watermark. This is a form of redundant embedding in order to improve the robustness of the watermark. When an image is distorted, not all coefficients in its representation are affected equally. Some coefficients may be affected more severe than others. One general strategy for the method to survive a wide variety of distortions is to embed a watermark redundantly across several coefficients. If some of these coefficients are damaged, the watermark in other coefficients should remain detectable.

Watermarks that are embedded redundantly by tiling can be detected in a number of ways. One method is to combine data from all the tiles in an image and then decode the result and test for the watermarks in their average. Another method is to test for the watermark in each tile independently, announcing that the watermark is present if

it is detected in more than some fraction of the locations. In our method the latter one is used. As long as one watermark survives the distortion, we assume that the watermark is detectable.

After all the watermarks are embedded in the Fourier transform, there is still one thing left before the embedding process is completed. An inverse Fourier transform has to be performed to map the signal back from the frequency domain into the spatial domain:

$$f(i, j) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k, l) e^{i2\pi(\frac{k}{N} + \frac{l}{N})} \quad (18)$$

As soon as the inverse DFT is computed, we can get the watermarked image. The operating result of the embedding process is shown in Figure 13. The original and the watermarked images are shown in Figure 14. As we can see, the watermarks are quite invisible.

```

ca 管理员: C:\Windows\system32\cmd.exe
D:\Study\研究生课程资料\MDM\R&D Project\Project\testCU\bin\Debug>testCU.exe

4 features have been found.
4 features left for watermarking.

=====
===== Choose The Generator =====
=====

[1] Random noise generator      [2] Chaotic sequence generator

Enter Choice: 2

Using chaotic sequence generator...

Crown: min: 0.000000, max: 1.000000
NoisyCrown: min: -4.984695, max: 4.994945
Fourier Real part: min: -1587119.500000, max: 32386628.000000

PSNR: 43.783773 dB

Coordinates of the 4 Keypoints:
127.016975      329.812836
384.983032      182.187180
255.872772      466.819366
256.127228      45.180634
  
```

Figure 13. Screenshot of the operating result of embedding process



Figure 14. The original image (a) and the watermarked image (b)

PSNR is used as an objective metric to evaluate the perceived quality of the watermarked image in addition to human eye observation. The PSNR is defined as:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX^2}{MSE} \right) \quad (19)$$

where  $MAX$  denotes the highest value in an image, and  $MSE$  denotes the mean squared error, which is defined as:

$$MSE = \frac{1}{mn} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} [I_w(x, y) - I(x, y)]^2 \quad (20)$$

where  $m$  and  $n$  are the dimensions of an image,  $x$  and  $y$  are the coordinates of an image,  $I_w$  is the watermarked image and  $I$  is the original image.

Because many signals have a very wide dynamic range, the value of PSNR is usually expressed in terms of the logarithmic decibel scale. Values above 40 dB indicate good quality, while values below 30 dB indicate low quality, the higher the better. Therefore, we consider the watermark to be quite invisible if the PSNR value is above 40 dB.

## 2.4 Watermark Detection

When the embedding process is completed, a detection process has to be proceeded to determine whether the watermark is detectable after the image has gone through common distortions or intentional tampering.

The general scheme of our detection process can be summarized as follows. First take the watermarked (and probably distorted) image, perform discrete Fourier transform on it and compute the magnitude coefficients. When the magnitude is obtained, run SURF on it to find new interest points. Since the image is watermarked and distorted, the interest points found by SURF are likely to be slighted different from the ones found during embedding. After we get the new interest points, instead

of embedding watermarks, we extract the square regions around these points in the real part of Fourier spectrum, the size of which equals to the diameter of the watermark. These regions are considered to be the possible regions that contain the watermarks. The normalized cross correlation between them and the original watermark is computed to determine if the watermarks are detectable or not. Also, the Grubbs' test is implemented to compute the number of outliers in the correlation maps. It is used as an alternative to a detection threshold. The block diagram of this process is shown below:

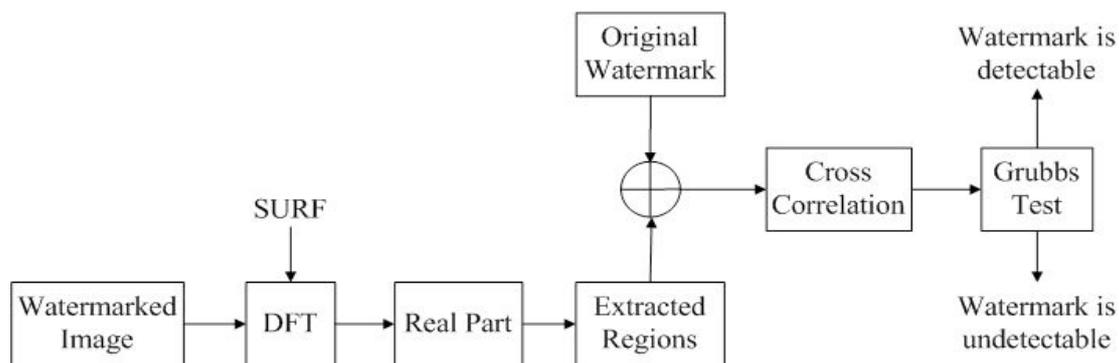


Figure 15. Block diagram of our detection process

The first two steps in detection are the same with the embedding. The only difference is that this time the watermarked image is used. Since the detector we implemented is a blind detector, we do not need the initial unwatermarked image, only the watermarked one. This will not doubt improve the adaptability of our proposing method because it is not constrained by the presence of the original image. In some applications where the original image is not available, non-blind detector is of no use, while blind detector can work perfectly fine.

Therefore, we take the watermarked image generated in embedding process and perform DFT on it. After splitting the real part and imaginary part of the Fourier transform, the magnitude coefficients are computed. Then SURF is applied again in the magnitude coefficients to find new interest points. Because the image is watermarked and possibly distorted, this will impose changes in the Fourier transform, causing SURF to find interest points that are shifted from the original ones found during embedding, or find completely different points. Since we do not know which points are the actual embedding locations, we have to consider all points to be potential embedding locations and save them all.

For example, the image below is a comparison between the interest points find by SURF with the original *Lena* image (during embedding) and the watermarked image without attacks (during detection). The white points are found during embedding, while the pink points are found during detection. We can see that the two points in the first and third quadrants are slightly shifted while the two points in the second and fourth quadrants are completely different. But as long as we can find some of the original points or points close to the original ones, the detector is still working.

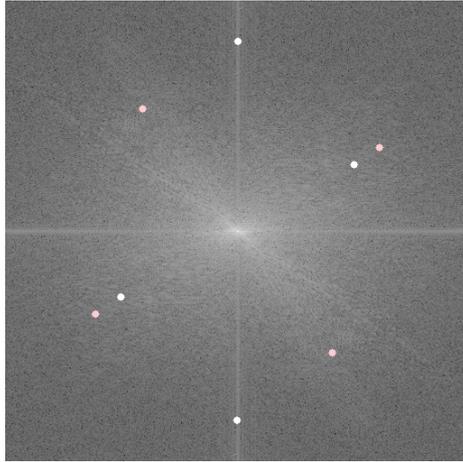


Figure 16. Interest points found during embedding and detection

After the new interest points are found, the next step is to extract the potential watermarked regions. Since the embedding is done in the real part of the Fourier transform, the extraction also needs to take place in the real part. Once the coordinates of the new interest points are loaded, we go into the real part and start the extraction. The size of the extracted square regions equals to the diameter of the watermark, the center of those regions are the coordinates of the interest points. As mentioned before, the Fourier transform has symmetry property, so the spectrum itself and the interest points found by SURF are both symmetric. Hence, we only need to save half of regions in the third and fourth quadrants, because the ones in the first and second quadrants are the symmetric version of these ones, and the correlation will be exactly the same. These regions are saved as text image for further use. The two images below are the two regions extracted in the third and fourth quadrants of the real part of the watermarked *Lena* image without attack. The left one contains a large portion of the watermark, so it is the right watermarked region. The right one contains no watermark, so this region is not watermarked.

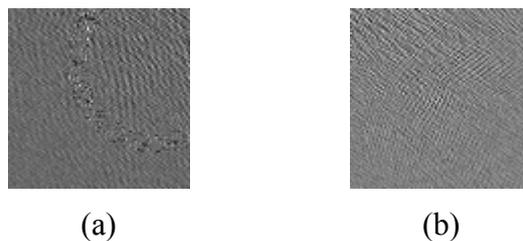


Figure 17. The extracted regions of the real part of Fourier transform

The reason we extract these regions is to calculate the similarities between them and the watermark, in order to determine if the watermark is robust enough and can still remain detectable even after distortions. This way, we do not need to go through the whole real part to compute similarities at every point, but rather target on the regions that have the biggest probabilities of containing the watermarks, which will make it faster to compute and easier to find peaks in the correlation maps.

There are many possible measurements to assess the similarity between two

images, and in our case, we use the normalized cross correlation. Cross correlation is a standard method of estimating the degree to which two series are correlated. It is a measure of similarity. For image-processing applications in which the brightness of the image and template can vary due to lighting and exposure conditions, the images should be first normalized. This is typically done at every step by subtracting the mean and dividing by the standard deviation. The value of the normalized cross correlation will vary between -1 and 1. A value of 1 indicates that at that point, the two images have the exact same shape, while a value of -1 indicates that they have the same shape except that they have the opposite signs. A value close to 0 indicates that they are uncorrelated.

To identify the maximum correlation, we have to compare the original watermark against the extracted region by sliding it. By sliding, we mean moving the patch one pixel at a time, left to right, up to down. At each location, a metric is calculated so it represents how good or bad the match at that location is, or how similar the watermark is to that particular area of the extracted region. For each location, the result is stored into a two dimensional array called correlation map. Therefore, the height of the correlation map equals to  $2*h-1$ , and the width equals to  $2*w-1$ , where  $h$  and  $w$  are the height and width of the watermark respectively. When all the results are calculated and saved into correlation map, the largest one in the map is the maximum correlation, which means at that point, the similarity between the extracted region and the watermark is the highest. If we take a look at the correlation map, we can see that the brightest location indicates the highest match, while the dark region indicates low similarity. The following image is the correlation map for watermarked *Lena* image with 1 degree of rotation, and the bright point marked by the red circle is the maximum correlation.

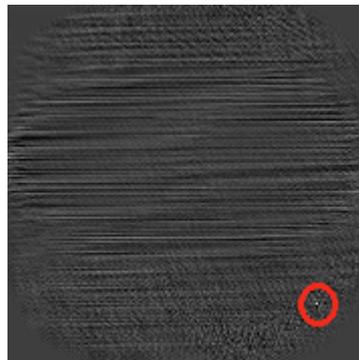


Figure 18. The correlation map

We have also tried the template matching function to compute the correlation, which is a powerful function provided by OpenCV. However, it does not work as well as the function we implemented. Because unlikely the function we use, template matching can only slide the template image within the boundary of the source image. If the extracted region contains a small portion of the watermark, while our method can detect it, template match cannot because the watermark cannot slide outside the extracted region. These two ways of sliding are illustrated below:

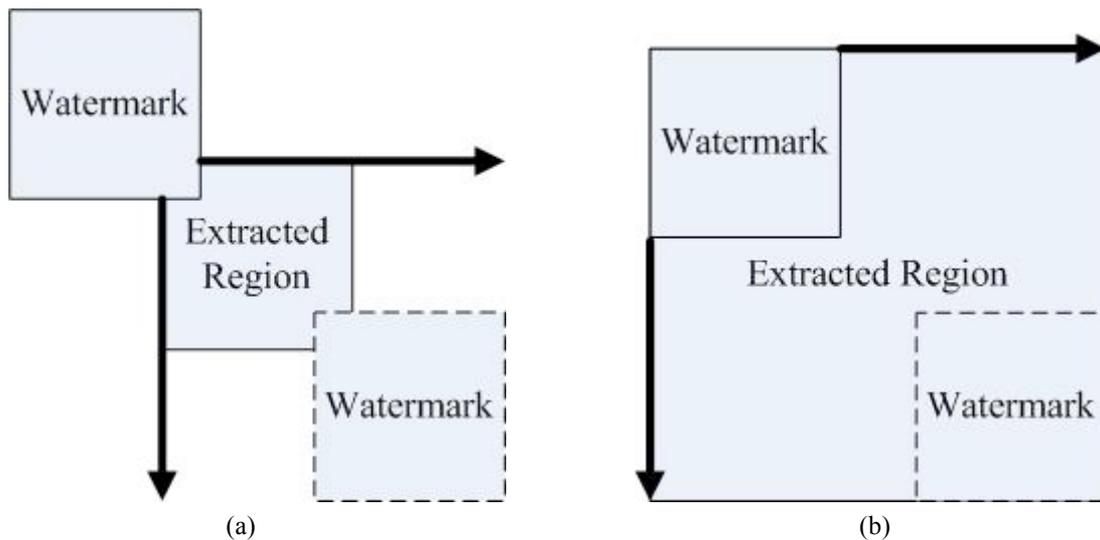


Figure 19. The two ways of sliding: (a) Implemented method; (b) Template matching

Be aware that the maximum correlation for our method may not be very high since we implemented a blind detector which do not requires the original unmarked image. Without the original image, we cannot extract the watermark, only the regions that may contain the watermark. So we have to correlate the original watermark which ranges between -1 and 1 with the extract region of real part whose range is tens of thousands times larger. Therefore, the decrease of correlation is to be expected. This is not a big problem, as long as the maximum correlation can be separated from the rest of the values. In other words, there should be a clear peak in the correlation map. The image below is the 3D plot of the correlation map shown in Figure 18. As we can see, the peak is clearly separated from other values. In cases like this, we can conclude that the watermark is detectable.

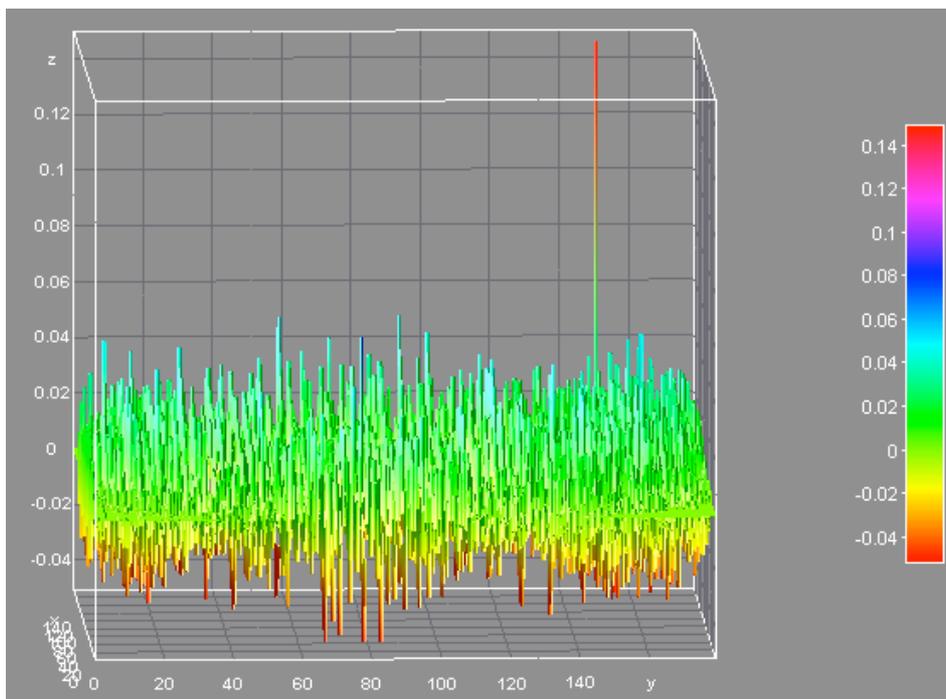


Figure 20. 3D plot of correlation map

Until now, we can only determine whether the watermark is detectable by viewing the correlation map and check if there is a peak. However, it would be too much trouble to do it for every region we extracted and for every image. So, it would be nice to implement an auto-detection function to tell us if there is a peak in the correlation map.

Commonly, most watermarking methods will set a value  $T$  as the threshold during detection. When the maximum correlation is returned, they compare the maximum value with the threshold  $T$ . If the maximum value is above  $T$ , the watermark is detectable. If not, the watermark is undetectable. This is the usual way to determine the presence of the watermark. The problem is how to choose the appropriate  $T$  value. Most methods simply use an empirical value obtained by conducting experiments on the test images. While it works well for this specific image dataset, it may not be suitable for other datasets with completely different image structure. So, instead of using a fixed threshold, we choose to use Grubbs' test to detect outliers, which is image dependent and highly adaptable. As long as we can find a clear peak (an outlier) in the correlation map, we consider the watermark is detectable.

Grubbs' test is a statistical test used to detect outliers in a univariate data set that follows an approximately normal distribution. It is also known as the maximum normed residual test. Grubbs' test detects one outlier at a time. This outlier is expunged from the dataset and the test is iterated until no outliers are detected.

Grubbs' test is defined for the hypothesis:

$H_0$ : There are no outliers in the data set

$H_a$ : There is at least one outlier in the data set

The Grubbs' test statistic is defined as:

$$G = \frac{\max |Y_i - \bar{Y}|}{s} \quad (21)$$

where  $\bar{Y}$  and  $s$  denote the sample mean and standard deviation, respectively. The Grubbs' test statistic is the largest absolute deviation from the sample mean in units of the sample standard deviation.

This is the two-sided version of the test. The Grubbs' test can also be defined as one of the following one-sided tests:

1. Test whether the minimum value is an outlier:

$$G = \frac{\bar{Y} - Y_{\min}}{s} \quad (22)$$

with  $Y_{\min}$  denoting the minimum value.

2. Test whether the maximum value is an outlier:

$$G = \frac{Y_{\max} - \bar{Y}}{s} \quad (23)$$

with  $Y_{\max}$  denoting the maximum value.

For the two-sided test, the hypothesis of no outliers is rejected at significance level  $\alpha$  if:

$$G > \frac{(N-1)}{\sqrt{N}} \sqrt{\frac{(t_{\alpha/(2N), N-2})^2}{N-2 + (t_{\alpha/(2N), N-2})^2}} \quad (24)$$

with  $t_{\alpha/(2N), N-2}$  denoting the critical value of the  $t$  distribution with  $(N-2)$  degrees of freedom and a significance level of  $\alpha/(2N)$ . For the one-sided tests, replace  $\alpha/(2N)$  with  $\alpha/N$ .

The significance level  $\alpha$  (different from the embedding strength  $\alpha$  in embedding process) is the criterion used for rejecting the null hypothesis. It is the probability of incorrectly rejecting a given null hypothesis in favor of a second alternative hypothesis. In our method,  $\alpha$  is set to 0.01, which means the false positive rate is controlled at 0.01.

Since the Grubbs' test can only be applied on a Gaussian distributed data set, and the correlation map does follow an approximately Gaussian distribution, therefore we apply Grubbs' test on the correlation map to find the number of outliers in it. If the number of outliers is more than 1, we conclude that the watermark is detectable. If not, the watermark is undetectable. The following plot is the histogram of the correlation map shown in Figure 18. We can see that it indeed follows a Gaussian distribution.

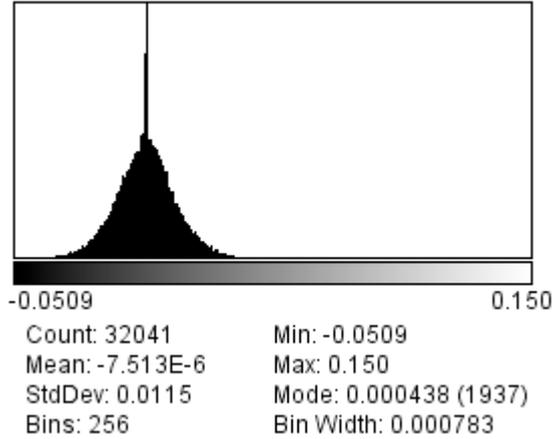


Figure 21. Histogram of the correlation map

The operating result of the detection process is shown below. As we can see, for the two regions we extracted before (see Figure 17, one contains the watermark, one does not), the one containing the watermark returns a relatively high maximum correlation with two outliers, one outlier value equals to the maximum correlation; while the one that does not contain the watermark returns a low maximum correlation with no outliers. Therefore, we conclude that the watermark in the first region is detectable while the watermark in the second region is undetectable.

```
ca. 管理员: C:\Windows\system32\cmd.exe
D:\Study\研究生课程资料\MDM\R&D Project\Project\testDetect\bin\Debug>testDetect.exe

=====The Maximum Correlation For Region 1=====
Coordinate: <99.687561,348.901031>
Height = 90, Width = 90
Max Correlation: 0.268563

-----

Number of Outliers: 2
Position of Outliers:
3633
12827

Value of Outliers:
-0.058387
0.268563

=====End=====

=====The Maximum Correlation For Region 2=====
Coordinate: <360.101563,391.846771>
Height = 90, Width = 90
Max Correlation: 0.051060

-----

Number of Outliers: 0
Position of Outliers:
NULL

Value of Outliers:
NULL

=====End=====
```

Figure 22. Screenshot of the operating result of detection process

### 3. Experimental Results

In this section, we will first test a variety of parameter combinations to decide an optimal one. Then, experiments are conducted to test the robustness and invisibility of the proposed method. Results are also provided to demonstrate that our method has considerable robustness against rotation, noise, Gaussian blur and JPEG compression while maintaining a relatively good perceived quality of the image.

#### 3.1 Parameter Selection

In our method, there are mainly five parameters that require tuning, the radius of the watermark (*radius*), the thickness of the watermark (*thickness*), the radius of the inner filter (*filter\_in*), the radius of the outer filter (*filter\_out*), and the embedding strength  $\alpha$ .

The first two parameters together determine the size of the watermark. Since our watermark is a crown watermark, the parameter *radius* denotes the distance from the center of the circle to the middle of the ring, while *thickness* denotes the thickness of the ring. Therefore, the radius of the inner ring is  $radius\_in = radius - thickness/2$ , and the radius of the outer ring is  $radius\_out = radius + thickness/2$ . This is illustrated below:

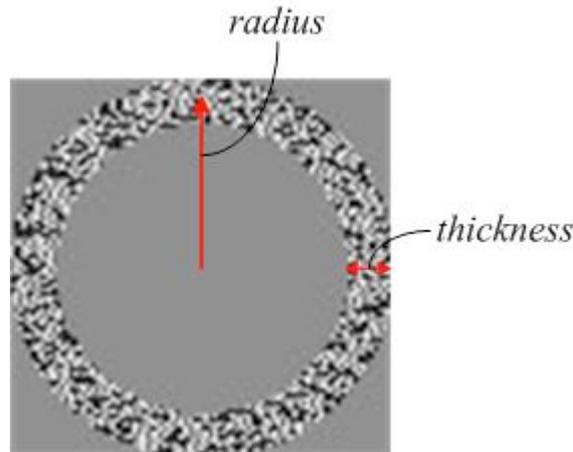


Figure 23. Parameters of *radius* and *thickness*

These two parameters need to be tested because if the watermark is too small, it may change the property of the interest point and cause SURF to lose this point during detection, which affects the robustness; if the watermark is too big, it will impose more changes on the Fourier transform and cause the PSNR to decrease, which affects the invisibility.

The next two parameters *filter\_in* and *filter\_out* determine the embedding area. Since we want to embed in the middle frequencies of the Fourier transform, we have to set up two filters to eliminate the interest points found in low and high frequencies. The parameter *filter\_in* denotes the radius of the low frequency filter, and *filter\_out*

denotes the radius of the high frequency filter. This is illustrated below:

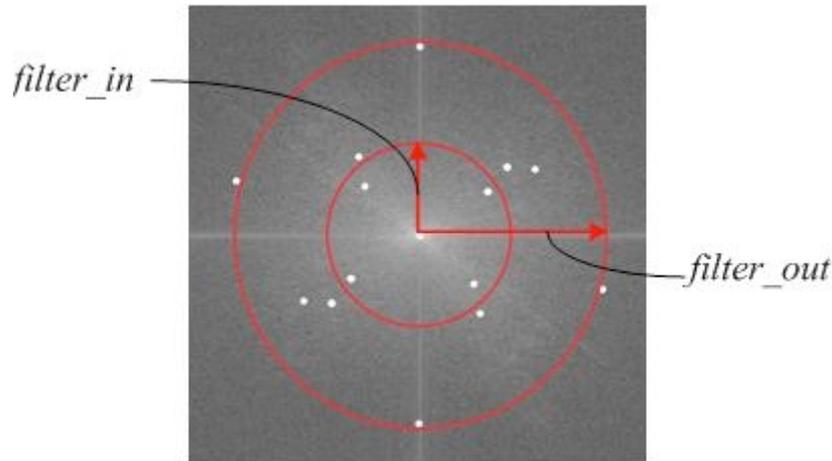


Figure 24. Parameters of *filter\_in* and *filter\_out*

These two parameters need to be tested because we want to give SURF enough space to find interest points while keeping these points away from the low and high frequencies, since modifications in the low frequencies will affect the invisibility and modifications in the high frequencies will affect the robustness.

The last parameter  $\alpha$  determines the extent to which the watermark alters the image. It needs to be tested because if it is too small, the watermark will not be robust enough; if it is too big, the watermark will not be invisible enough.

We have tested 20 different parameter combinations in total on the *Lena* image with both the chaotic generator and random generator. With each combination, the PSNR of the watermarked image is recorded. The watermarked image is then attacked by: rotation of  $0.5^\circ$ ,  $1^\circ$ ,  $1.5^\circ$ ,  $2^\circ$ ,  $2.5^\circ$  and  $3^\circ$  respectively; Gaussian noise with a mean of zero and a standard deviation of 5 and 10 respectively; Gaussian blur with a standard deviation of 1; JPEG compression with a compression ratio of 70%. After each attack, the correlation is recorded, along with the correlation when there is no attack.

During the experiments, we confirmed that as the embedding strength increases, the PSNR decreases while the correlation increases. Also, as the embedding locations shift towards higher frequencies, the PSNR increases while the correlation decreases. As for the comparison between the chaotic generator and the random generator, the PSNRs are very close with the same parameters, while the correlations are inconclusive. For some parameter combinations, the chaotic generator outperforms the random generator against certain attacks, while for other parameter combinations the random generator outperforms the chaotic generator for certain attacks. But despite the differences in maximum correlation values, the two generators can both detect the watermarks under most circumstances. And since the chaotic generator is more secure than the random generator, we decide to choose the former one.

Figure 25(a) is the average PSNRs of different embedding strength ( $\alpha = 3, 4, 5, 6, 7$ ) for both chaotic and random generators. As we can see, the PSNR drops as the embedding strength increases.

Figure 25(b) is the average PSNRs of different embedding areas and the corresponding parameters are shown below, where  $w$  represents the width of the image and  $radius\_out$  represents the radius of the outer ring of the watermark.

| tick label | $filter\_in$   | $filter\_out$                        |
|------------|----------------|--------------------------------------|
| 1          | $\frac{1}{4}w$ | $\frac{1}{2}w - 2 \cdot radius\_out$ |
| 2          | $\frac{1}{4}w$ | $\frac{1}{2}w - radius\_out$         |
| 3          | $\frac{1}{3}w$ | $\frac{1}{2}w - radius\_out$         |

Table 2. Corresponding parameters used in Figure 25(b)

The drop in the middle is because SURF finds more interest points in this area. So it embeds more watermarks than the other two using this parameter combination, which will no doubt deteriorate the image quality. And in both plots, the PSNRs of random generator are slightly higher than chaotic generators, but the differences are very small. Also, the average PSNRs are all above 40, which indicate good perceived quality.

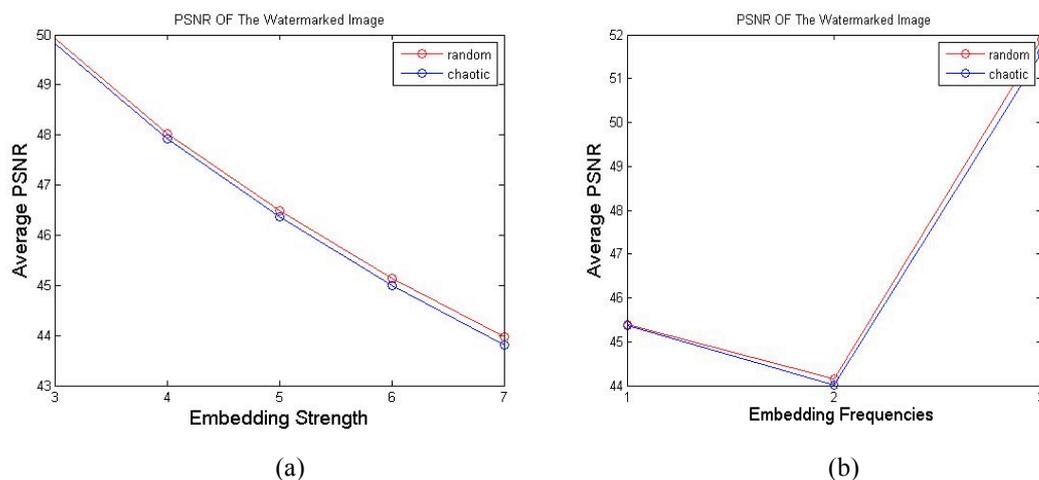
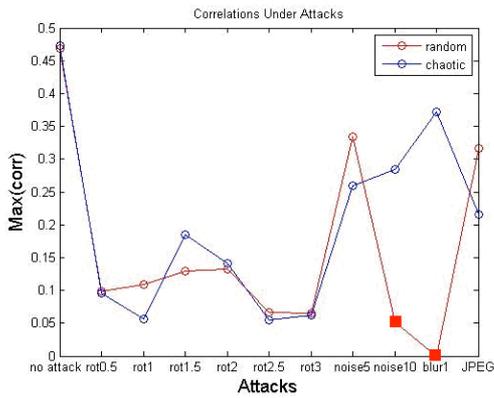


Figure 25. Average PSNRs of the *Lena* image under different circumstances

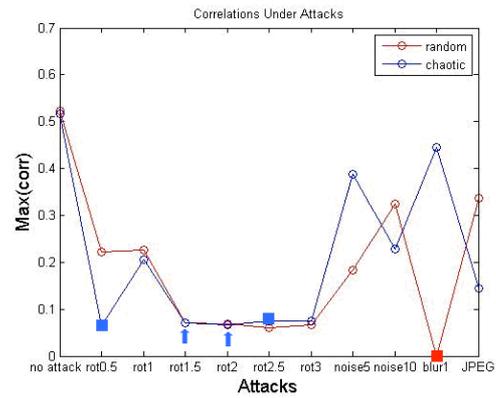
In Figure 26 we have displayed six plots of the correlations of a watermarked image under different attacks. The image used here is the *Lena* image. These are only part of the combinations we have tested (6 out of 20), displayed in order to explain how we choose the optimal parameter combination. A hollow circle means the detection is successful, and a filled square or an arrow means the detection is unsuccessful. The parameter combinations we use in the following plots are as follows:

| No. | $radius$ | $thickness$ | $filter\_in$   | $filter\_out$                        | $\alpha$ |
|-----|----------|-------------|----------------|--------------------------------------|----------|
| (a) | 40       | 10          | $\frac{1}{4}w$ | $\frac{1}{2}w - 2 \cdot radius\_out$ | 5        |
| (b) | 30       | 10          | $\frac{1}{4}w$ | $\frac{1}{2}w - 2 \cdot radius\_out$ | 5        |
| (c) | 40       | 10          | $\frac{1}{3}w$ | $\frac{1}{2}w - radius\_out$         | 5        |
| (d) | 40       | 10          | $\frac{1}{4}w$ | $\frac{1}{2}w - radius\_out$         | 5        |
| (e) | 40       | 10          | $\frac{1}{4}w$ | $\frac{1}{2}w - radius\_out$         | 4        |
| (f) | 40       | 10          | $\frac{1}{4}w$ | $\frac{1}{2}w - radius\_out$         | 6        |

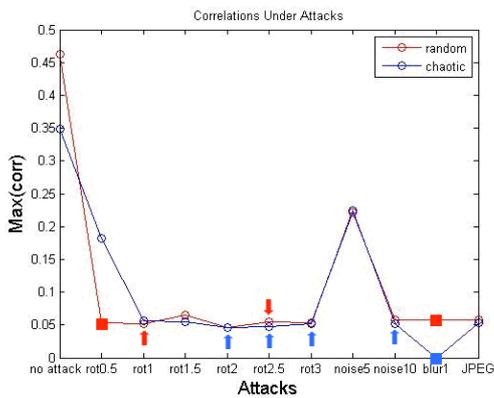
Table 3. Parameter combinations



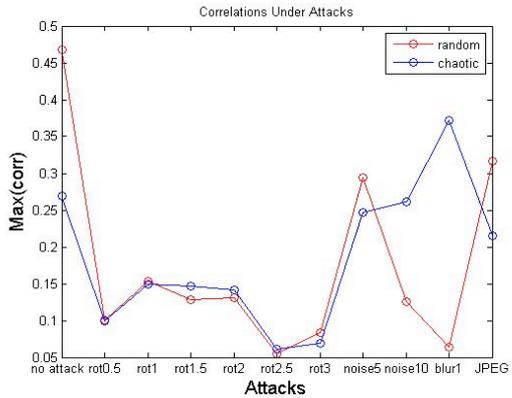
(a)



(b)



(c)



(d)

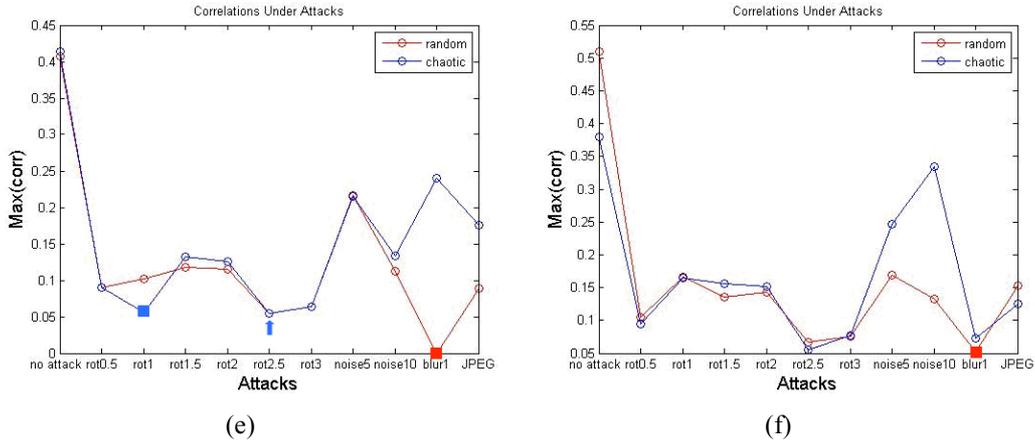
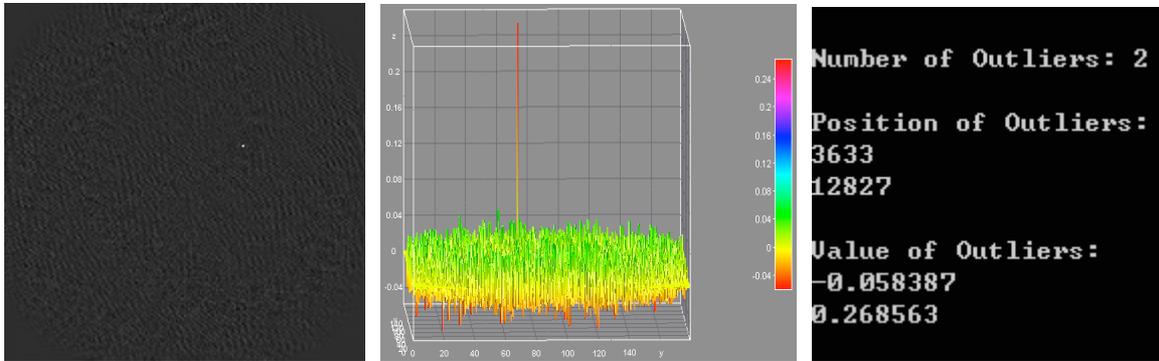


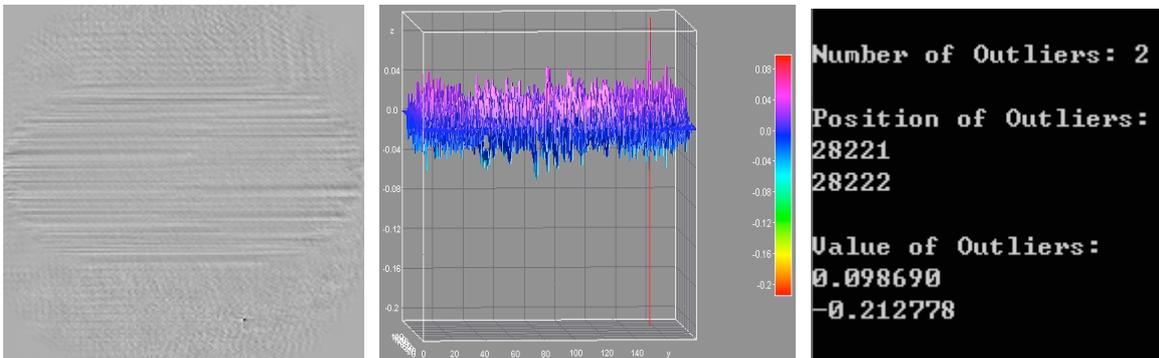
Figure 26. Correlations of the *Lena* image under different circumstances

The difference between (a) and (b) is the size of the watermark, as we can see, if the size decreases, the robustness against rotation decreases as well. The difference between (c) and (d) is the embedding frequencies, and as the embedding locations move towards higher frequencies, the robustness against all attacks decreases accordingly. The difference between (e) and (f) is the embedding strength, and as the embedding strength decreases, the robustness against all attacks also decreases.

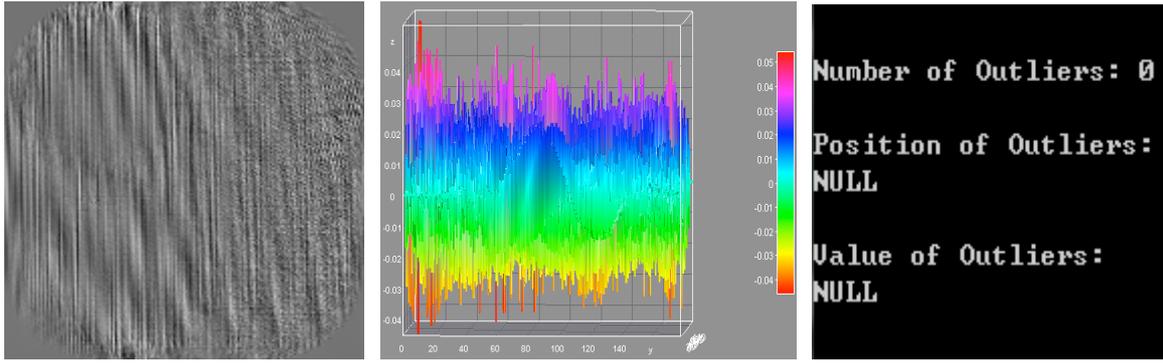
The results of the Grubbs' test are shown below. Figure 27(a) is a very successful Grubbs' test on image *Lena* without attack. Figure 27(b) is a mildly successful Grubbs' test on image *Lena* with 0.5 degree rotation. Figure 27(c) is an unsuccessful Grubbs' test on image *Lena* with 2.5 degree rotation.



(a) A very successful Grubbs' test



(b) A mildly successful Grubbs's test



(c) An unsuccessful Grubbs' test

Figure 27. Results of the Grubbs' test on image *Lena*

After considering the tradeoffs between the robustness and invisibility, we decide to use the following parameter combination:

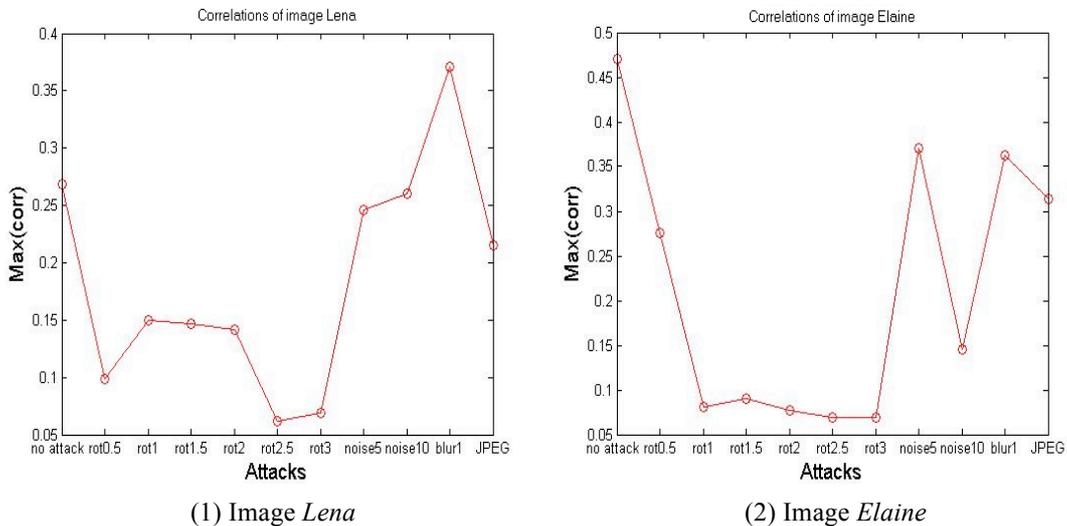
| <i>radius</i> | <i>thickness</i> | <i>filter_in</i> | <i>filter_out</i>            | $\alpha$ | generator |
|---------------|------------------|------------------|------------------------------|----------|-----------|
| 40            | 10               | $\frac{1}{4}w$   | $\frac{1}{2}w - radius\_out$ | 5        | chaotic   |

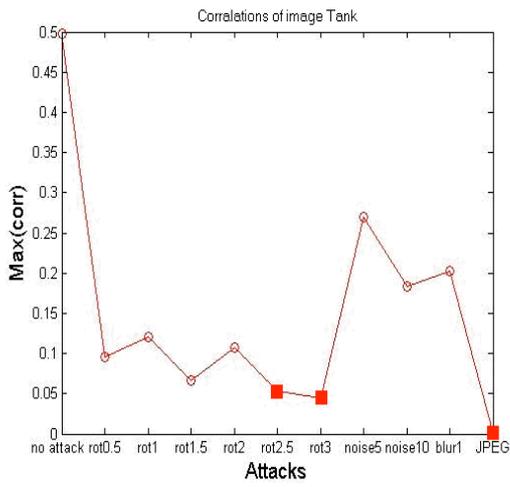
Table 4. The optimal parameter combination

We believe that with this parameter combination, the method can achieve a good robustness and invisibility at the same time.

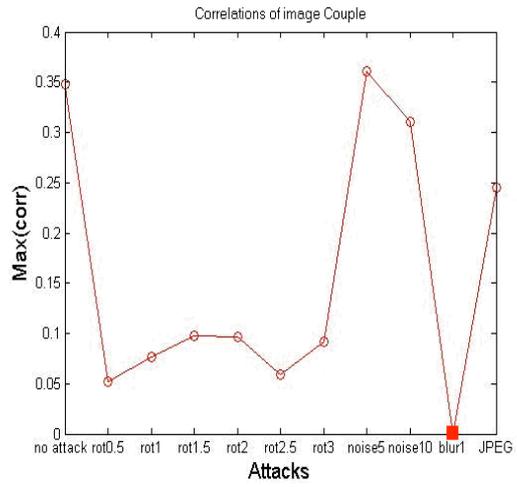
### 3.2 Experiment Results

After we determined the final parameter combination based on image *Lena*, we conducted a number of experiments on 10 different images with this chosen parameter combination to test the robustness and invisibility of the proposed method. The results are shown below:

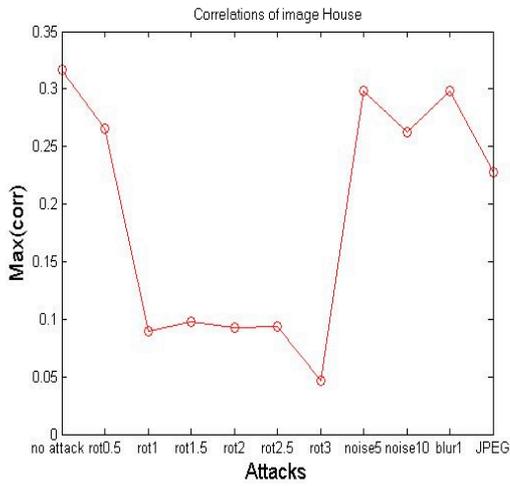




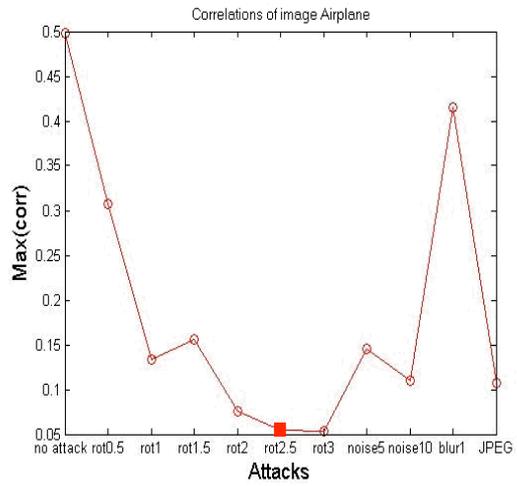
(3) Image Tank



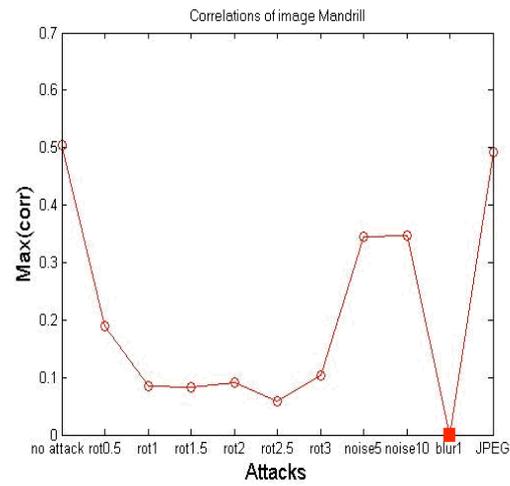
(4) Image Couple



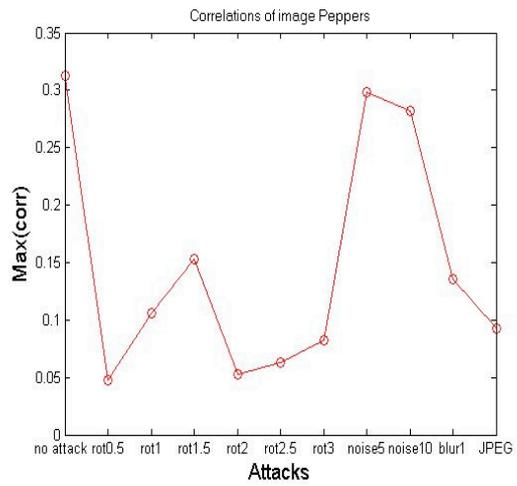
(5) Image House



(6) Image Airplane



(7) Image Mandrill



(8) Image Peppers

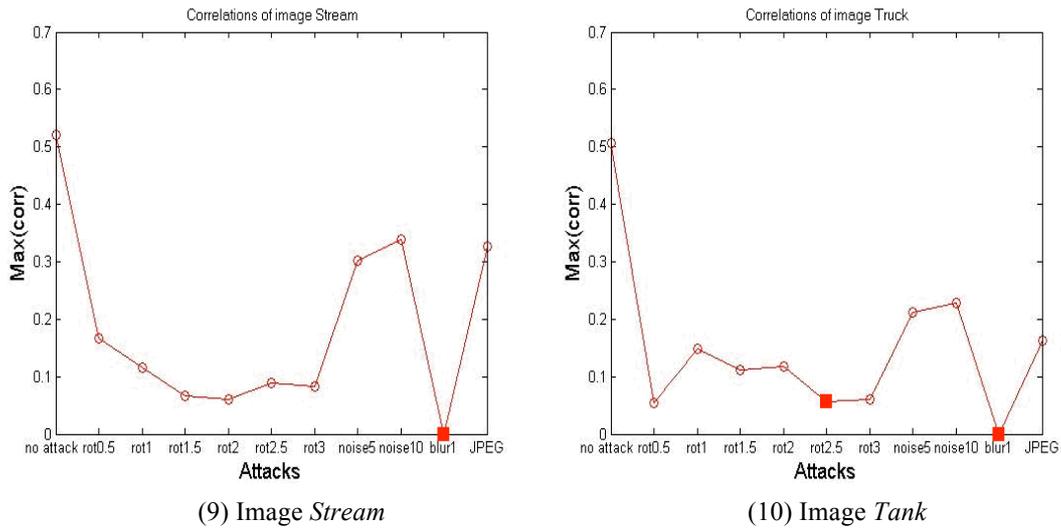


Figure 28. Correlations under attacks for different watermarked images

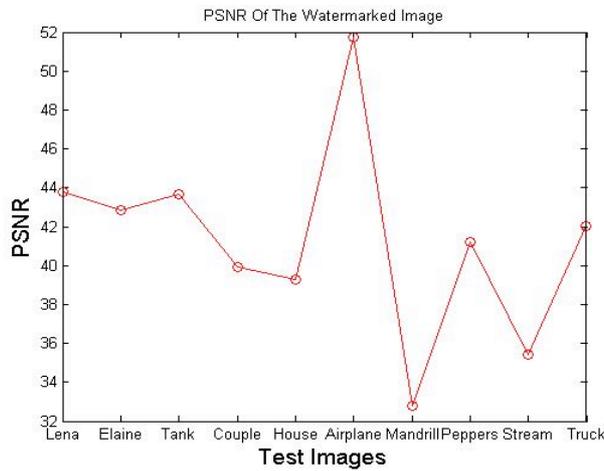


Figure 29. PSNRs of different watermarked images

As we can see from Figure 28, almost all the test images can resist up to 2 degree rotation, some can even resist to a 2.5 degree rotation while some of the watermarking approaches cannot even resist a 0.5 degree rotation. All the images can withstand noise attack, most of them can resist JPEG compression, and more than half of them can resist Gaussian blur. And the Grubbs' test can provide satisfying results for most of the tests. A hollow circle represents a successful detection, and a filled square represents an unsuccessful detection. Also, from Figure 29 we can see that almost all the test images have a PSNR above 40 after embedding, except *Mandrill* and *Stream*. These experiment results show that our proposed watermarking method has considerable robustness against different kinds of distortions while maintaining a good perceived quality.

It should be mentioned that there is not enough time for us to conduct experiments about the false positive probability and false negative probability. Such a study could have been useful to further compare the performance between the chaotic sequence generator and the random noise generator.

## 4. Conclusion

In this report we presented a novel watermarking method based on crown-shaped watermarks applied in the DFT domain. The embedding locations are found using SURF algorithm and the watermarks are generated using chaotic map. Multiplicative embedding is applied during the embedding procedure, and the original image is not required in the detection procedure. The watermarks are embedded in the middle frequencies of the DFT domain. PSNR is used to evaluate the perceived quality of the watermarked image. Normalized cross correlation is used to evaluate the similarity between the original watermark and the extracted regions containing the watermark. Grubbs' test is implemented to detect outliers in the correlation maps in order to determine whether the watermark is detectable.

Experimental results show that this method is robust against a variety of distortions, including rotation (up to 2 degree), Gaussian noise, Gaussian blur and JPEG compression. Also, the security level is improved due to the use of chaotic sequences to form the watermark.

There is still much room for improvement in future works. First, our method is only applicable to grayscale images, all the color images are converted to grayscale images before embedding. We can try to expand the scope of the method to color images. Second, although SURF works fine with the 10 images we tested, it may fail when applied to other images because it is originally designed to be used in spatial domain, not frequency domain. We can try to improve it or find new ways to find interest points in a highly noisy environment like DFT domain. Third, we can try to use 2D Gaussian function to smooth the edges, which may improve the performance of the method. Fourth, we can try to adjust the parameters dynamically with different images instead of applying one fixed set of parameters on every image. This will make the method more image dependant and achieve better robustness. Fifth, we can use some masking techniques to increase the invisibility of the watermarks. Sixth, conduct experiments about the false positive and false negative probabilities to further compare the performance of chaotic sequence generator with random noise generator.

## References

- [1] V. Solachidis, I. Pitas, "Circularly symmetric watermark embedding in 2-D DFT domain", *IEEE Trans on Image Processing*, 2001;10(11):1741-53.
- [2] V. Solachidis and I. Pitas, "Optimal detection for multiplicative watermarks embedded in DFT domain," *Proceedings 2003 ICIP*, vol. 2, p. 3, Sept. 2003.
- [3] I.J. Cox , J. Kilian , T. Leighton , T. Shamoan, "Secure Spread Spectrum Watermarking for Multimedia", *IEEE Transactions on Image Processing*, 6(12), pp. 1673-1687, 1997.
- [4] I. J. Cox, M. L. Miller. "A review of watermarking and the importance of perceptual modeling;" *Proc. Of SPIE Electronic Imaging*, February 1997.
- [5] J. O'Ruanaidh, T. Pun, "Rotation, scale and translation invariant digital image watermarking," *Signal Processing*, May 1998, Vol.66, No.3, pp.303-317.
- [6] A. Poljicak, L. Mandic, D. Agic, "Discrete Fourier transform-based watermarking method with an optimal implementation radius," *Journal of Electronic Imaging*, vol. 20, no. 3, p. 033008, Jul. 2011.
- [7] R. L. Tataru, S. El Assad, "Improved blind DCT watermarking by using chaotic sequences," *International Conference for Internet Technology and Secured Transactions*, pp. 46-50, Dec. 2012.
- [8] K. Solanki, U. Madhow, B. S. Manjunath, S. Chandrasekaran, "Estimating and undoing rotation for print-scan resilient data hiding," presented at *the ICIP*, Singapore, Oct. 2004.
- [9] K. Solanki, U. Madhow, B. S. Manjunath, S. Chandrasekaran, I. El-Khalil, "Print and Scan Resilient Data Hiding in Images," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 4, pp. 464-478, 2006.
- [10] S. Voloshynovskiy, A. Herrigel, N. Baumgaertner, T. Pun, "A stochastic approach to content adaptive digital image watermarking," in: *Third International Workshop on Information Hiding*, vol. 1768, pp. 212-236, 1999.
- [11] P. Le Callet, F. Autrusseau, P. Campisi, "Visibility Control and Quality Assessment of Watermarking and Data Hiding Algorithms," *Multimedia Forensics and Security*, Chapter IX, pp. 164-193, 2008.
- [12] R. Zhao, H. Lian H, H. Pang, B. Hu, "A Blind Watermarking Algorithm Based

on DCT,” *2008 Second International Symposium on Intelligent Information Technology Application*, vol. 3, pp. 821–824, 2008.

[13] I. J. Cox , M. Miller , J. Bloom , J. Fridrich , T. Kalker, *Digital Watermarking and Steganography*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2007.

[14] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, “Speed-Up Robust Features (SURF),” *Computer Vision and Image Understanding (CVIU)*, vol. 110, No. 3, pp. 346-359, 2008.

[15] F.E. Grubbs, “Procedures for Detecting Outlying Observations in Samples,” *Technometrics*, 11(1), 1969.