Blind Digital Watermark Based on Discrete Fourier Transformation

Kaixuan Zhang

Macau University of Science and Technology, 999078, Macao

Abstract. The policy and security of digital images are concerned by all artists. Embedding blind digital watermarks to images is an effective solution to such problem. The existing blind digital watermark techniques for images are usually based on the spatial domain method, such as some build-in tools in some software like Photoshop. Those methods' imperceptibility is good, but the robustness is dissatisfactory. Therefore, blind digital watermark based on the frequency domain is becoming popular nowadays. The use of the frequency domain-based techniques can not only keep the imperceptibility but also ensure good robustness and non-removability. According to the reasons mentioned above, this thesis presents a frequency domain-based blind digital watermark technique. This method uses basic discrete Fourier transformation to transform an image to the frequency domain and then embeds a digital watermark. To increase the security, we also introduce some methods to encode watermarks before embedding them to the images. We conduct some experiments by applying the technique to some images with different watermarks. In the experiments, it can embed blind watermarks with the least effect on the quality of the original image and can also successfully extract watermarks from the watermarked image. The results show this technique is highly useable. We also conduct some experiments to simulate the attacks. This technique also shows great robustness after many kinds of different attacking methods.

Keywords: Digital, Blind, Watermark, Encoding, Frequency, Domain, Fourier, Transformation.

1. Introduction

Digital art, including digital painting and digital images, is increasingly become mainstream in modern art appearance. How to protect the privacy of artists' work become a hit in the early years. To protect the copyright of the authors, we need to add something to the image work like a watermark to declare the ownership of the image. So, there was a lot of research on the privacy of digital watermarks for images in the last decades, However, with the development of the image process tools, typical watermarks become weaker to many kinds of attacks. A more imperceptible and non-removable watermark technique should be conducted.

Using blind watermark techniques can largely increase imperceptibility and non- removability. Blind watermark is the watermark that is invisible to human perception. It can be added without destroying the original work to achieve copyright protection and tracking. There are two main algorithms for the blind watermark. One is based on the spatial domain and the other is based on the transform domain algorithm. A typical method for spatial domain such as using image information hidden based on the least significant bit (LSB). And Patchwork algorithm is based on the change of the brightness of some pixels to hide the watermark. They all add an invisible blind watermark to the image with great imperceptibility.

All the above blind watermark techniques are added the watermark in the spatial domain. Though they all have great imperceptibility, the robustness is not that satisfactory. Some basic attacking methods such as Scaling attack, Shear attack, Rotation attack, and so on can easily damage the watermark or even remove the watermark. Under such circumstances, such a spatial domain algorithm is not suitable.

There needs a new blind watermark technique to increase the robustness. So, the watermark based on the transform domain is introduced for such conditions. Digital blind watermark based on frequency domain techniques uses some kind of transformation like discrete Fourier transformation (DFT), discrete Cosine transform (DCT), or discrete Wavelet transform (DWT), to transform the digital image to the frequency domain and add the blind watermark in the frequency domain. By

using the inverse transformation, we can convert the image back to a spatial domain with the blink watermark. The watermark is also invisible to human perception so it has great imperceptibility, and because it is hidden in the frequency domain, so the robustness is more acceptable. But according to the transform function and the algorithm embedding the watermark to the frequency domain, the robustness can be various.

This thesis will use the DFT as the transformation function and encode the watermark before embedding it in the frequency domain of the image. Our human is more sensitive to the change of the low frequency in an image. Encoding the watermark before embedding it to the frequency domain of an image will control the energy distribution of the watermark. It will do less influence on the energy distribution of the original image itself. This means we humans will harder to notice the change in the original image. And it can also encrypt the watermark, which can make the watermark safer. However, there are many challenges are faced in developing such a blind watermark approach.

First, how to create and encode the watermark? The kind of watermark is quite various. We need to set a standard for the watermark. Our goal is to control the energy distribution of the watermark, we need to introduce a convenient and efficacy encoding method to achieve that.

Second, how to embed a watermark to the frequency domain of images? Different algorithms for embedding the watermark to the image may influent the robustness and imperceptibility of the watermark. In some recent studies, we can see that the robustness will influent the imperceptibility and so does the inverse. They will influent each other, which means high robustness results in low imperceptibility. We need to balance them by adjusting our algorithm.

Third, how to get the watermark from the watermarked image? It not only depends on the algorithm when we add the watermark but also depends on the encoding method we choose when encoding the watermark. We can do an inverse operation if we successfully solve the first two challenges. This work addresses the above challenges and introduces a frequency domain blind watermark technique for the image. Basic knowledge about watermarks is learned from a reference book, this technique uses the Random sequence encoding method to encode the watermark. The seed for the random number can be random or got from the size of the original image, which makes it more convenient to decode and get the extracted watermark. We will use Two-dimensional Fast Fourier transform as the transform function to transform the image to the frequency domain and use the inverse version to transform it back. The Fast Fourier Transform is equivalent to the normal discrete Fourier transform, which is faster by means of butterfly merging. Different from the traditional blind watermark method, we regard the watermark as a kind of noise and add it to the frequency domain of the image. And we use a coefficient called energy coefficient E to control the energy of the watermark. It can not only guarantee a good imperceptibility of the watermark but is also robust to various common attacks.

2. The Process of Watermark

2.1 Creating Watermark

We input the watermark to MATLAB by using the LPT toolbox provided by MATLAB. LPT provides plenty of useful functions for image processing in MATLAB. We can simply use imread () function to input the watermark. And we can also input the image need to be watermarked to MATLAB at the same time for preparation. In order to simplify our following steps, we can normalize them by using mat2gray () function. But before that, we need to change the type of the data to double type because the type of most of the images is uint8. After these small preparations, we can begin to edit our watermark.

The size of the watermark we prepared will always not match the size of the image needed to watermark. Or we can say the size of the matrix of them is not the same. In order to successfully embed the watermark to the image, we need to create a new image with a watermark that the size is the same to the watermarked image. We can use size () function to get the size of the image and use zero () function to create a matrix or say an empty image. Here because we will encode and symmetric

the watermark in the following step, we can simply create half of the new watermark image in this step and do a symmetric copy after we encode the half. It will save some time in the encoding section. So, the size of the new watermark image in this step will be the same size of watermarked image's width in width and 50% size of its height in height. Then we can directly copy the watermark we prepared to the empty new image we create. Encoding the watermark has two obvious benefits. One is to encrypt our watermark. The attacker will not easily extract the watermark and remove it if he doesn't know the encoding method we choose. And the other is to control the energy distribution of the watermark. In general, the energy of one image is mainly distributed in the low-frequency part. And our human eyes are more sensitive to the change of the low-frequency part. So, when we embed the watermark to the image frequency domain, we need to avoid making too many changes to the low-frequency part of the image. In this way, we can make a trick that the image seems to have no change

The other one is to define a special seed as the random number generator seed. There are many ways to define the special seed. In my experiment, the special seed is simply generated by multiplying the width, the height, and the energy coefficient E of the watermark image. Energy Coefficient E is a constant set by the user. I will introduce the Energy Coefficient E in the following part. The reason is that if the seed is a random number, which means one watermarked image corresponding to one special encoding sequence. The encoder needs to save all of them in order to decode the watermark when he needs. It is very inconvenient. If the encoder lost the sequence by mistake, the encoder itself could not extract the watermark from the watermarked image. That is not what we want. So, I choose a more convenient to generate the random sequence number.

Input: M - the watermark, S - the seed to generate the random sequence, W - the width of the image, H - the half height of the image
Output: EM - the encoded watermark

```
RX := randX(seed);
RY := randY(seed);
for i = 1, 2, ..., H do
    for j = 1, 2, ..., W do
        EM(i,j,:) := M(RY(i),RX(j),:);
    end for
end for
```

Figure 1. Watermark Encoding

The first one is much safer compared to the other method. Because the key is only known by the encoder or say the author. But it's inconvenient to save the key. For the second method, if the seed generates method is acknowledged by the attacker, it will be easier for the attacker to get the key and try to remove the watermark. Each method is OK in this blind watermark technique. In my experiment, I will use the second encoding method.

2.2 Making Symmetric Watermark

The symmetric watermark can increase the robustness of the watermark. The first reason is that making symmetric watermark will increase the size of the watermark. The larger the watermark is, the more effective will make on the frequency, but the robustness will also increase. And when the attacker uses some attacks like cutting attacks to break the watermark, it will be not easy for the attacker to break it without hurting the quality of the image. Because we not only have an upper watermark but also at the lower part. We can extract the watermark on the other side if one side is broken by the attacker. After successfully encoding the upper half of the watermark image we can use a small loop sentence to make a symmetric copy of the lower half. And we create a full-size empty image that is the same size as the watermarked image by using the zeros () function and directly copying the upper and the lower half of the encoded watermarked to it. Then we get the full encoded

watermark prepared to embed to the watermarked image. As shown in Figure 2, the figure gives a sample of the encoded watermark.

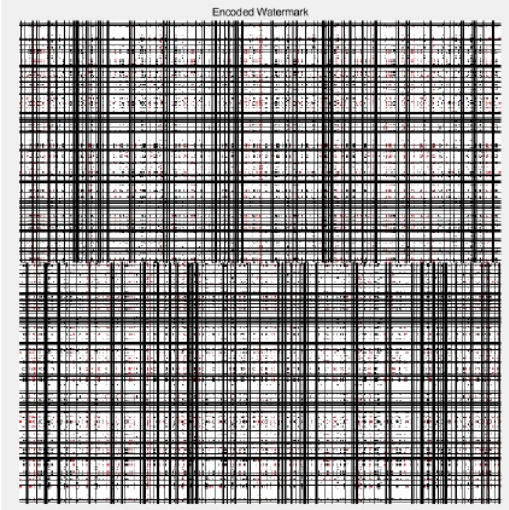


Figure 2. Sample of Encoded Watermark

3. Embedding Watermark

3.1 Overview of Embedding Watermark

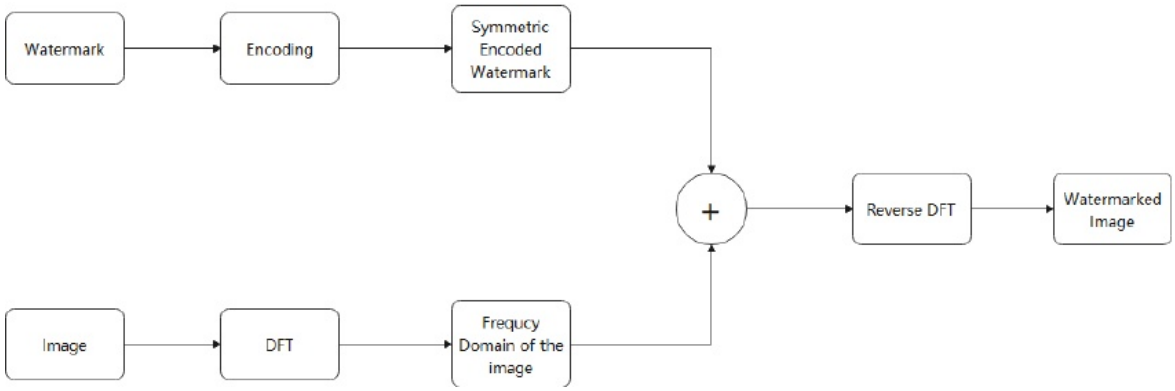


Figure 3. Flowchart of Embedding the Watermark

Figure 3 shows the process of embedding the watermark. The upper half is the process for the watermark, which we have mentioned in Chapter 2. And in this Chapter, we will introduce the other part of the flow chart one by one, which is the main procedure of embedding the watermark.

After we understand the energy coefficient E and choose the exact value we need, we can embed the watermark and use inverse Two-dimensional Fast Fourier transform to transform the watermarked image to the spatial domain.

$$I_m = ifft2(fft2(im) + E * emark) \quad (1)$$

Equation 1 shows the full procedure of embedding the watermark, variable im is the original image, we transform it to the frequency domain by using the build in function fft2(). And emark is the encoded mark we got at the previous step. After setting the energy coefficient E, we multiply it to emark and embed it to the frequency domain of the original image. Then we just need to transform the image back to the spatial domain and the result Im is the watermarked image we want. There are also build-in function ifft2() in MATLAB for us to use.

What need to notice is the result of ifft2 often have the imaginary part. To save and show our result, we can use real () function to get only the real part of the watermarked image. As shown in Figure 7, the spectrum of the original image and the watermarked image do have such a huge difference. However, there may have some small noise in the watermarked image, but it is quite tiny that the user

will not notice that easily. It is because we encoded the watermark before embedding it to the watermarked image, which makes the energy of the watermark evenly distributed in the watermarked image. Or we can say the noise is evenly distributed in every part of the watermarked image. So that's why the difference in the frequency domain is quite huge but the difference in the spatial domain is not that noticeable.

4. Extracting Watermark

4.1 Overview of Extracting Watermark

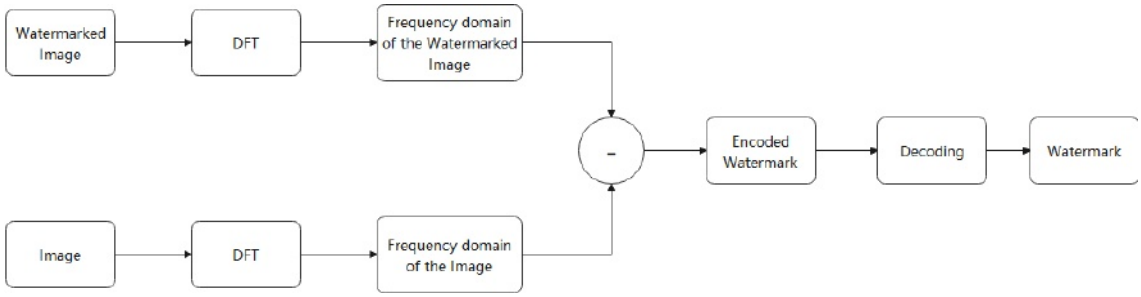


Figure 4. Flowchart of Extracting the Watermark

As shown in Figure 4, this is the flow chart of the extracting watermark process. It is quite similar to the process of embedding a watermark. We will introduce the procedure one by one in this chapter.

The watermarked is added in the frequency domain of the watermarked image. So, we need to transform the watermarked image and original image to the frequency domain. We will also use the Two-dimensional Fast Fourier transform which is the fft2() function in MATLAB to do that. Then, we use a simple minus operator (-) to minus the original image from the watermarked image, for the watermark is added as additive noise to the watermarked image frequency domain. The remaining part is the encoded watermark after the minus operation. Equation 2 shows the process of getting the encoded watermark Me. Variable mim means the watermarked image and the im is the original image.

$$M_e = fft2(mim) - fft2(im) \tag{2}$$

Lastly, we decoded the encoded watermarked by using the key or say the number sequence. It can be easily implemented by some loop statement. The pseudocode is shown in Algorithm 2. The process is the reverse process of the encoding, the only difference is we make the random position pixel to the correct place they should be according to the key or say random sequence we generate.

Input: EM - the encoded watermark, S - the seed to generate the random sequence,
W - the width of the image, H - the half height of the image
Output: M - the extracted watermark

```
RX := randX(seed);
RY := randY(seed);
for i = 1, 2, ..., H do
    for j = 1, 2, ..., W do
        M(RY(i),RX(j),:) := EM(i,j,:);
    end for
end for
```

Figure 5. Watermark Decoding

We only need to decode the upper part of the watermark and directly do a symmetric copy to the lower part. Because we did the same thing in the encoding part. Then we can get the symmetric watermark we prepared before but in the same size as the watermarked image. As shown in Figure 9, It shows that the extracted watermark is visible and of good quality so the artist can use it to show the copyright of the image.

5. Experimental Results and Discussion

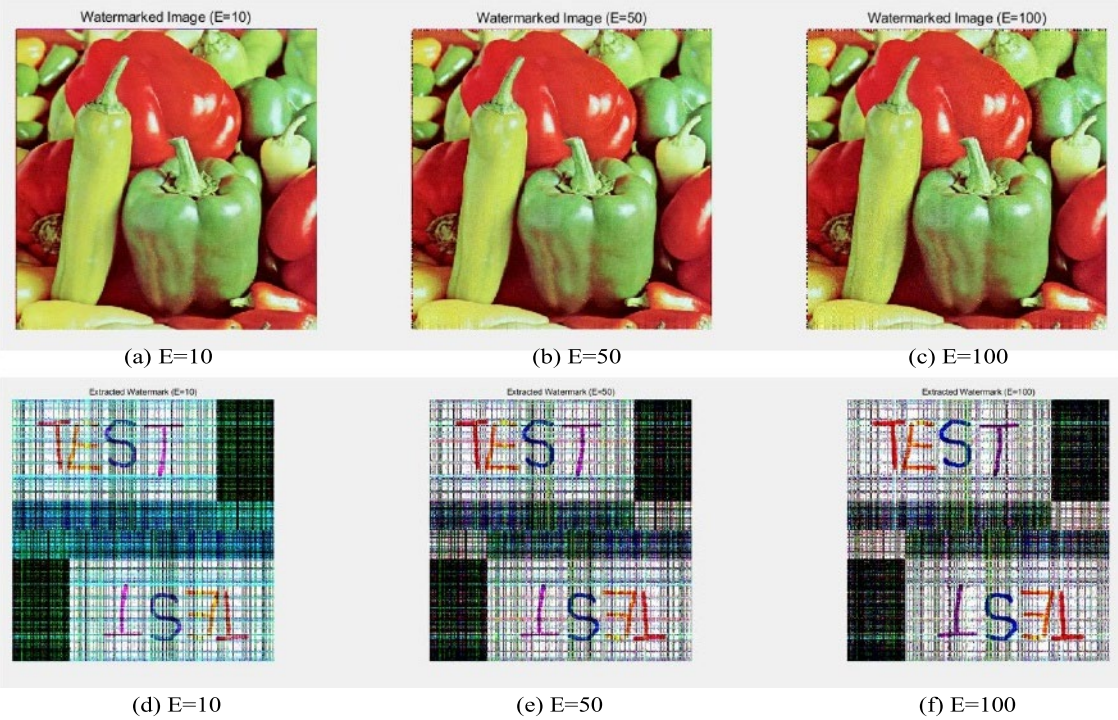
Imperceptibility and robustness are two key attributes of a usable blind watermark technique. We conducted some experiments on images provided by SIPI Image database to evaluate mainly the imperceptibility and robustness of our technique. And we also conduct some experiments compared with other blind techniques.

5.1 Imperceptibility

I will use Peppers.tiff from SIPI Image database as our watermarked image and a watermark drawn in Window Painting by myself as the watermark in this part, as shown in Figure 6.

(a) Original Image “Peppers” (b) Watermark

Figure 6. Test Image and Watermark



(a) E=10 (b) E=50 (c) E=100

(d) E=10 (e) E=50 (f) E=100

Figure 7. Watermarked Image (a), (b), (c) and Extracted Watermark (d), (e), (f) with different E

I will evaluate the imperceptibility by calculating residual, PSNR, and MCE, and the influence of energy coefficient on imperceptibility. The energy coefficient will be set to 10, 50, and 100 when embedding the watermark. And the encode method is using the special seed one. As shown in Figure

7, it shows the result of the watermarked image with this energy coefficient E, and also it shows the watermark extracted from the watermarked image when the E is different.

We calculate the MSE and PSNR for the watermarked image with different energy coefficients E. As shown in Table 1, the MSE and PSNR are quite small even if the energy coefficient is equal to 100, only in 65.7db. That means the difference between the original image and the watermark image is quite small in value. The result can show the great imperceptibility of our technique.

Table 1. The MSE and PSNR for Different E

Case	Energy coefficient E	MSE	PSNR
1	10	1.7479e-4	85.7 db
2	50	0.0044	71.3 db
3	100	0.0175	65.7 db

5.2 Robustness

In this part, I will simulate some common attacks on the watermarked image and try to extract the watermark to check if the watermark is in good condition or not. Some of the attacks are referred from a recent work. The attacks will be simulated by using tools like Windows Painting, MATLAB, and so on. I will use the same image Pepper as the watermarked image for the experiment. The encoding method is the same as in the previous experiment. And I will also use different energy coefficients (E = 10,50,100) to show the impact of the energy coefficient on the robustness of the blind watermark.

5.2.1 Smear Attack

I use Windows Painting and Photoshop to smear the watermarked image. All the 3 images are smeared with the same image. As shown in Figure 8, the watermark is still visible in all three watermarked images, but the quality is quite different. We can see only the outline of the watermark when the energy coefficient is equal to 10, but we can see clearly when the energy coefficient is equal to 100. But we still can conclude that the watermark has good robustness to smear attack.

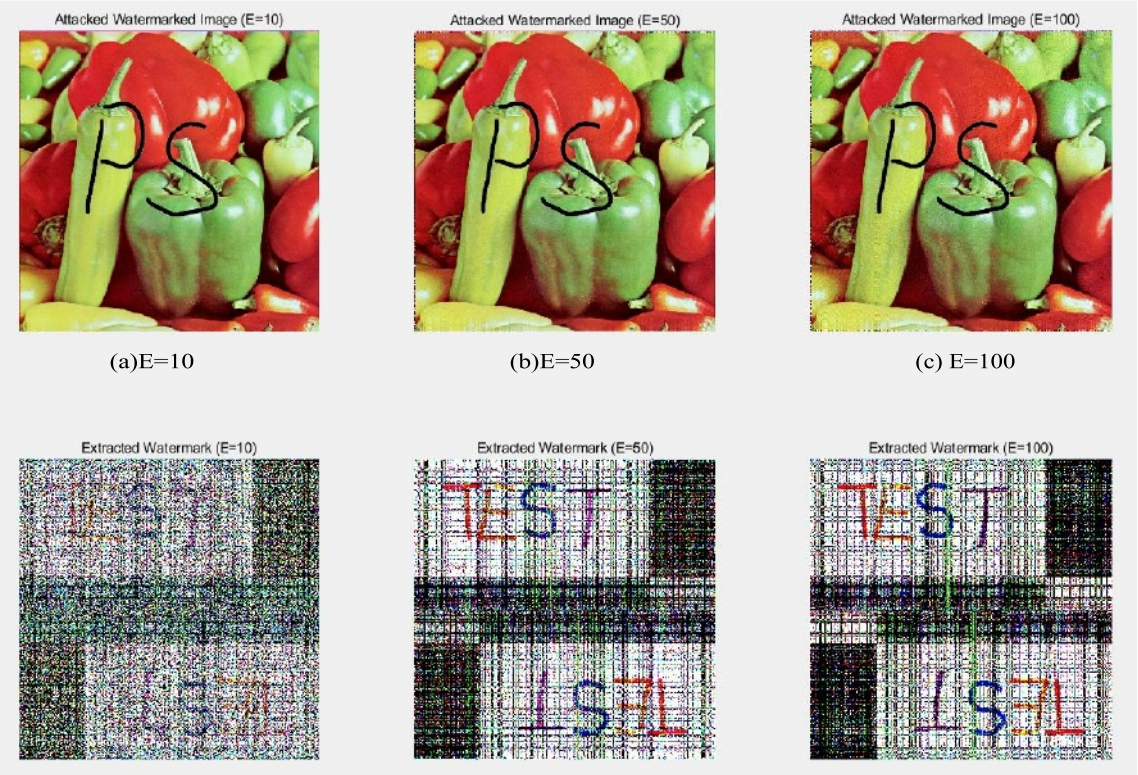


Figure 8. Extracted Watermark after Swear Attack

5.2.2 Rotation and Cutting Attack

The tool I use for rotating the watermarked image is Photoshop. I rotate the watermark 45 degrees clockwise to simulate the rotation attack. As shown in Figure 14, the four edges of the attacked watermarked image seem like cut by the attacker. In such conditions, we can regard such rotation also as a cutting attack. So that is why I put these two attacks in a part. When it turns to the result extracted watermark, we can see that in the condition of E=10, the watermark is already invisible. But as the energy coefficient increase, the watermark shows up in the condition of E=50 somewhere bad quality. So, we can conclude that if the user wants to improve the robustness to rotation and cutting attack, the energy coefficient should be set a little higher to make the watermark's quality better.

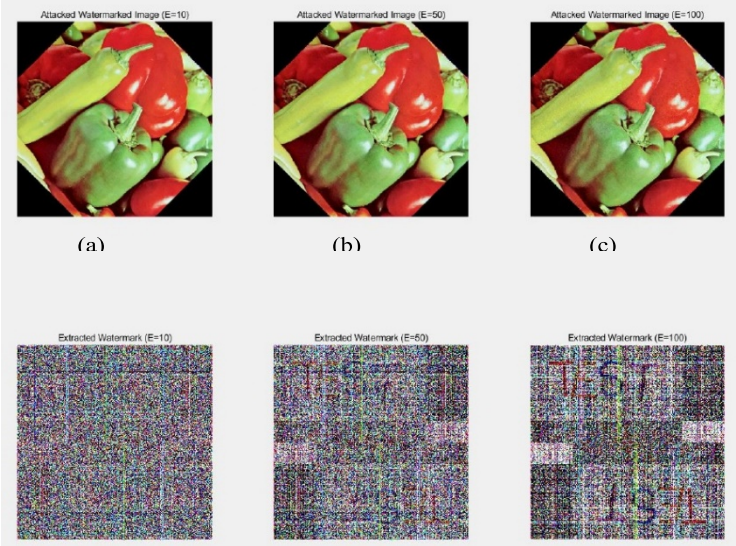


Figure 9. Extracted Watermark after Rotation Attack

5.2.3 Scaling Attack

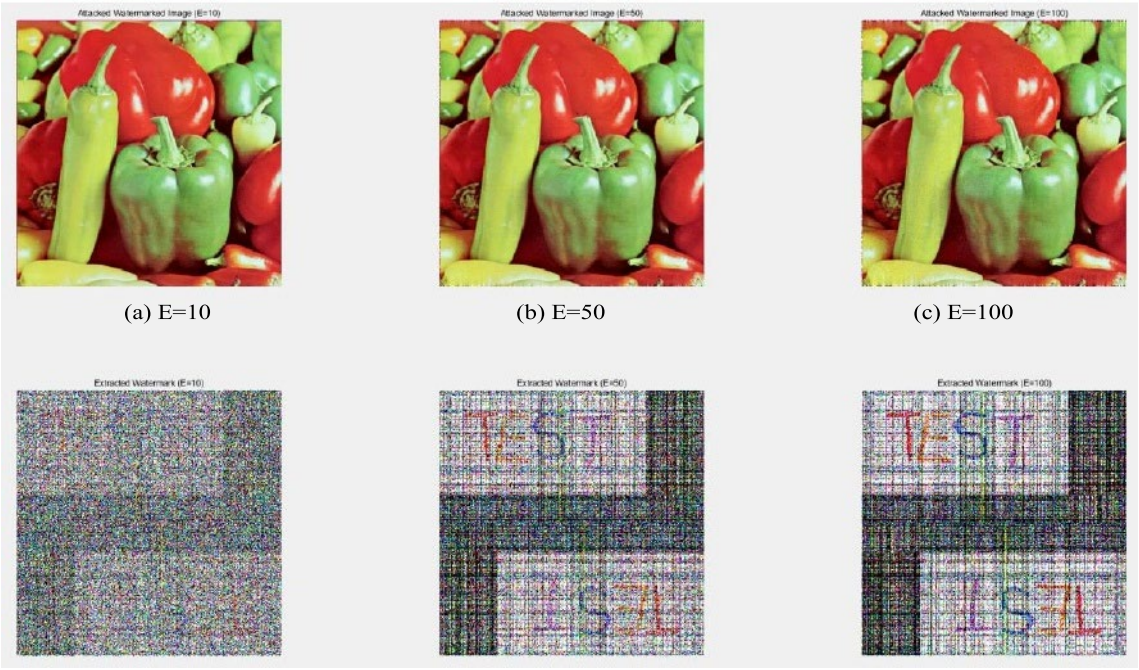
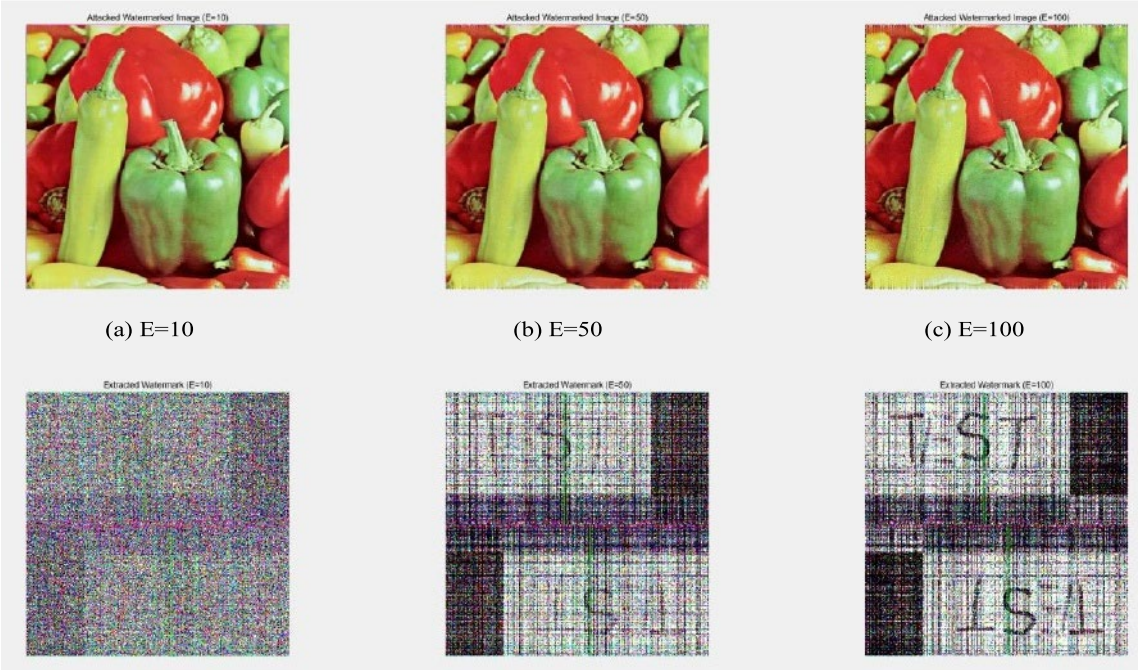


Figure 10. Extracted Watermark after Scaling Attack

In this part, I use MATLAB to simulate the Scaling Attack. Before saving the watermarked image, we can call imresize () to change the size of the output image. In this experiment, I set the output size to 0.8x, which means the output watermarked image is 0.8 times in size of the original watermarked

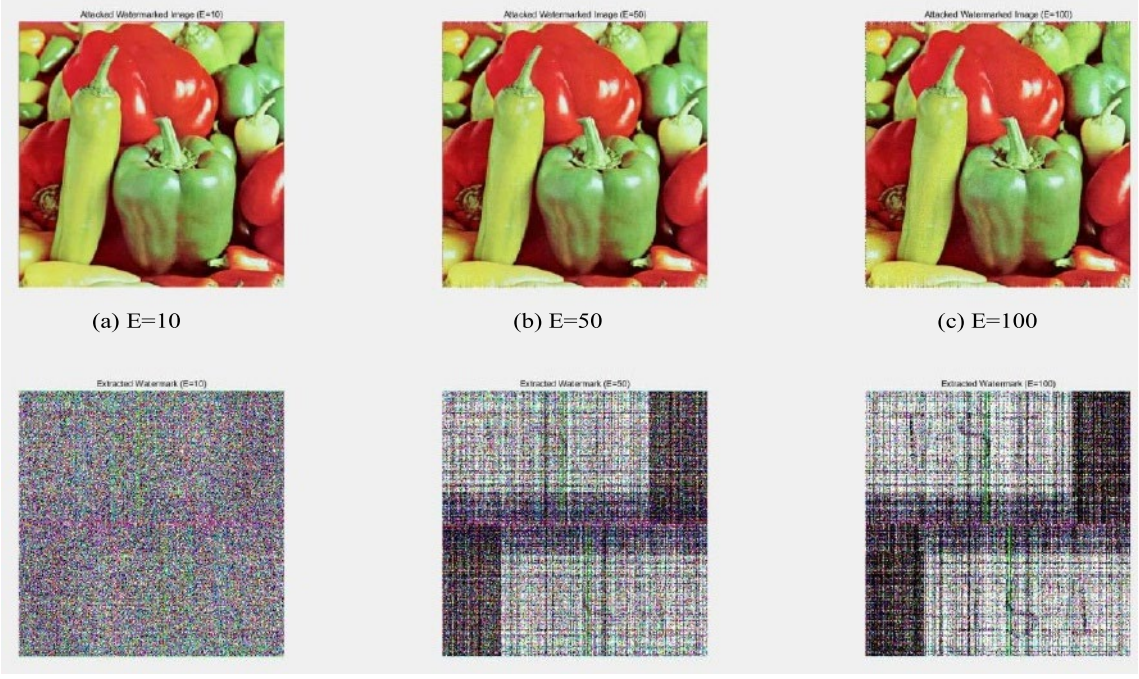
image. The interpolation method is a bicubic interpolation. Then we try to extract the watermark from them. As shown in Figure 10, the result is the watermark in E=10 watermarked image is of bad quality, but we still can find the outline of the watermark. While in the condition of E=50 and E=100, the quality of the watermark is quite better. So, we can see that the technique has good robustness to scaling attack.

5.2.4 Compressing Attack (JPEG)



(a) E=10 (b) E=50 (c) E=100

Figure 11. Extracted Watermark after Compressing Attack (JPEG 75%)



(a) E=10 (b) E=50 (c) E=100

(d) E=10 (e) E=50

Figure 12. Extracted Watermark after Compressing Attack (JPEG 40%)

JPEG compress is a compressing technique that is widely used all over the internet. It can significantly decrease the size of the image and do minimal damage to the image. However, it is still a type of lossy compression, which means it may damage our blind watermark hidden in the image. In this part, we will use JPEG compress to test the robustness of the compressing attack. In MATLAB,

we can choose the type of image to save on our disk. Jpg is also included. So, we can use MATLAB as our test tool. When you choose to save the image as a jpg file, you can define the quality of the image. The smaller the value you set, the worse the quality of the image will be. I set the quality of the jpg file to 75% and 40% to get the watermarked image and try to extract the watermark. As shown in Figure 11, when the quality is set to 75%, we can see that though we can see the white background of the watermark in the E=10 image, the content of the watermark is broken. While in E=50 and E=100, we can find the watermark, but the watermark is also damaged to a different extent. The situation is much worse in 40% quality, the watermark seems broken in the E=10 image. And we can only see some outlines in E=50 and E=100. We can conclude that our technique has acceptable robustness to JPEG compressing attack.

5.2.5 Image Sharpening

Image sharpening is a technique to make a blurred image clear. Some of the implementations are like a kind of high-pass filter, which means the low frequency of the image may be lost in such a procedure. I use Photoshop as the tool to attack the watermarked image. There is an image sharpening filter in Photoshop, and I apply it to the three watermarked images. As shown in Figure 13, we can see that the watermark is of good quality no matter if the E is set to 10 or bigger. We can conclude that the technique has good robustness to the image sharpening attack.

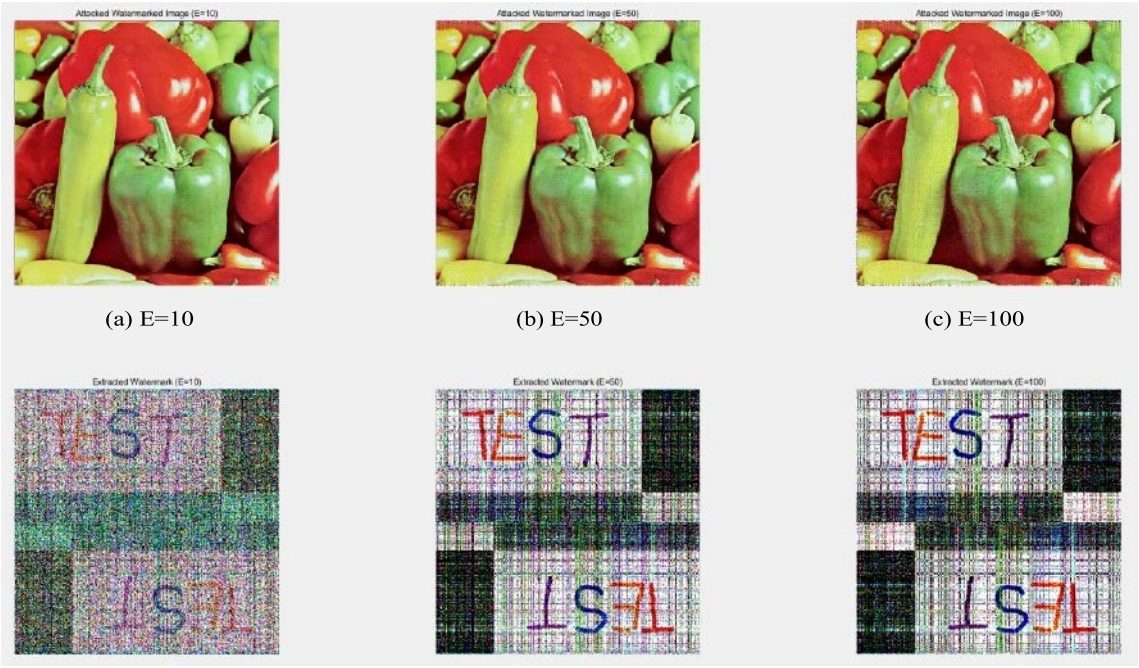
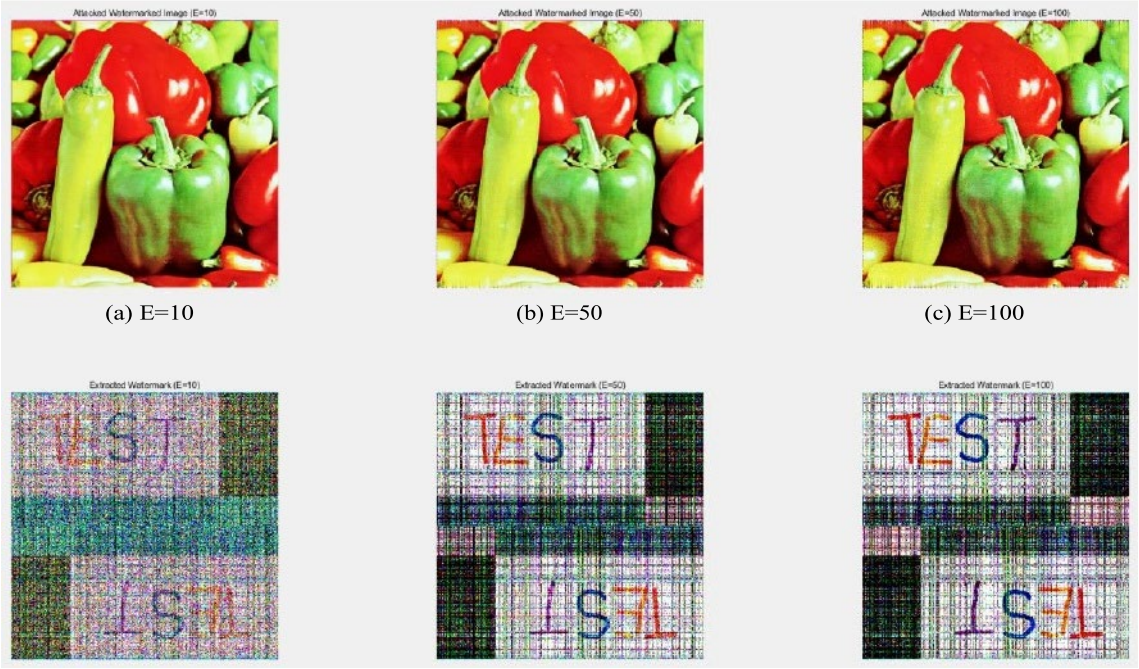


Figure 13. Extracted Watermark after Image Sharpening

5.2.6 Contrast Enhancement

Contrast enhancement can make the image more colorful. I also choose Photoshop as my tool to simulate the attack in this part. There is a function to change the contrast of the image. The default value is 0, and I set it to 100 for all test watermarked images. As shown in Figure 14, we can see that the contrast of the watermarked image is quite high. But the watermark extracted from them is quite clear. So, we can say that the technique has good robustness to the contrast enhancement attack.



(a) E=10 (b) E=50 (c) E=100

(d) E=10 (e) E=50 (f) E=100

Figure 14. Extracted Watermark after Contrast Enhancement

6. Conclusion

6.1 Conclusion

We can conclude from the experiments that our DFT technique shows a great imperceptibility and robustness for the blind watermark. The difference between the watermarked image and the original image is quite small and human eyes will not notice it easily. The thief will never know he gets the image with the watermark. We test many kinds of attacks to the watermark in this thesis and find that the technique shows great robustness to most of the attacks. We can successfully extract the watermark from the image after most of the attacks, even though the quality of the watermark is quite different.

However, the imperceptibility and robustness of digital blind watermarking are mutually exclusive. The difference between the watermarked image and the original image becomes obvious as the energy coefficient E get bigger, but the robustness is increasing at the same time. We can choose a reasonable value of E to balance them and satisfy the need of the artist.

6.2 Future Work

There are also many weaknesses in the technique. First, the robustness of the compressing attack and cutting attack is not satisfactory. Second, the encoding method for the watermark is also not convenient. Third, the procedure of embedding the watermark to the image is quite simple. And last, we need the original image when extracting the watermark.

In the future, I will try to improve the watermark encoding method by using a better algorithm to make it not only convenient when decoding, but also much safer compared with the current method. I will also try to find a more stable way to embed the watermark into the image. These will also be helpful to the improvement of the robustness and imperceptibility of the blind watermark. After all these improvements are done, I will make an application with GUI to make the procedure of embedding and extracting watermark much more convenient for the normal user to use. I plan to continuously improve the technique to address the weaknesses in future work.

References

[1] Akansu, A. N., Delp, E., Kalker, T., Liu, B., Memon, N., Moulin, P., & Tewfik, A. (2003). Special issue on signal processing for data hiding in digital media and secure content delivery. IEEE Transactions on Signal Processing, 51(4), 897-897.

[2] Barten, P. G. (2003). Formula for the contrast sensitivity of the human eye. In Image Quality and System Performance, 5294, 231-238

[3] Bors, A. G. and Pitas, I. (1996). Image watermarking using DCT domain constraints. Proc. of ICIP'96, 3, 231–234.

[4] Cayre, F., Fontaine C., and Furon T. (2005). Watermarking security: Theory and practice. IEEE Trans. Signal Process., 53(10), 3976–3987.

[5] Dugad, R., Ratakonda, H. and Ahuja, N. (1998). A new wavelet-based scheme for watermarking images. Proc. ICIP'98, 2, 419–423.

[6] Ejima, M. and Myazaki, A. (2001). On the evaluation of performance of digital watermarking in the frequency domain. Proc. of the IEEE on Image Processing, 2, 546-549.

[7] Guo, L. (2008). Research on image digital watermarking technology. Xi 'an Polytechnic University, 29, 19-20.

[8] Gonzalez, R. and Woods, R. (2008). Digital image processing. Harlow: Prentice Hall.

[9] Huang, Z. (2008). Digital watermarking algorithm based on LSB and its implementation in MATLAB. Modern Machinery, 2, 67-70.

[10] Jiwu Huang, Weidong Cheng (2000). Image watermarking in DCT domain: Embedding strategies and algorithms. Chinese Journal of Electronics, 28, 57-60

[11] Hu yan, Chen Zhao (2006). Application of MATLAB in digital Watermarking. Information Technology, 4, 184-186.

[12] Ingemar, J. C. and Miller, M. L. (2002). The first 50 years of electronic watermarking. Journal of Applied Signal Processing, 2, 126-132.

[13] Luo jianjun (2005). MATLAB tutorial. Publishing House of Electronics Industry.

[14] Petitcolas, A. P., Anderson R. J., and Kuhn, M. G. (1998). Attacks on copyright marking systems. Inform. Hiding, 1525, 218–238.

[15] Roberts, L. G. (1962). Picture coding using pseudo-random noise. IRE Trans. Inf. Theory, 8, 145-154.

[16] Sun shenghe, Lu Zeming, Niu Xiamu (2004). Digital watermarking technology and its application. Science Press.

[17] Solachidis, V. and Pitas, I. (2001). Circularly symmetric watermark embedding in 2-D DFT domain. IEEE Trans. Image Process, 10(11), 1741–1753.

[18] University of Southern California. (1977). The USC-SIPI Image Database. SIPI Image Database, Sipi.usc.edu/database/database.php

[19] Xing xiu-Qin, Ye Zhi-zhong, Xing Xiu-juan (2013). Digital Watermarking algorithm based on energy analysis. Digital Technology and Applications 9, 113-120.

[20] Yeo, I. K. and Kim, H. J. (2003). Generalized patchwork algorithm for image watermarking. Multimedia Systems 9, 261–265.