

SUSTech_EE326_Project_Final_Report

11911521钟新宇

1. Introduction

With the development of the Internet and social media, digital information can be easily copied and spread, leading to the security issues of digital information transmission and copyright protection of digital products. Digital watermarking is a technology that hides information into the carrier, and it is difficult for people to perceive the change of the carrier after the watermark is embedded, and they cannot directly access the watermark information. After the digital watermark is embedded, the carrier can be used normally. Compared with visible and perceivable information transmission solutions such as QR codes, digital watermarking is better hidden, more beautiful, and safer [1].

This project implements embedding and extraction of color image watermark based on spatial domain digital watermarking technology. We use baker transform to encrypt the watermark, and use lsb algorithm to embed the encrypted watermark information into the carrier. The embedding of the watermark does not affect the visual quality and integrity of the carrier. However, the lsb algorithm, as the most primitive digital watermarking technique, has poor robustness. It only has good anti-interference effect on pretzel noise, and cannot effectively deal with the attacks of Gaussian noise, histogram equalization, median filtering and mean filtering.

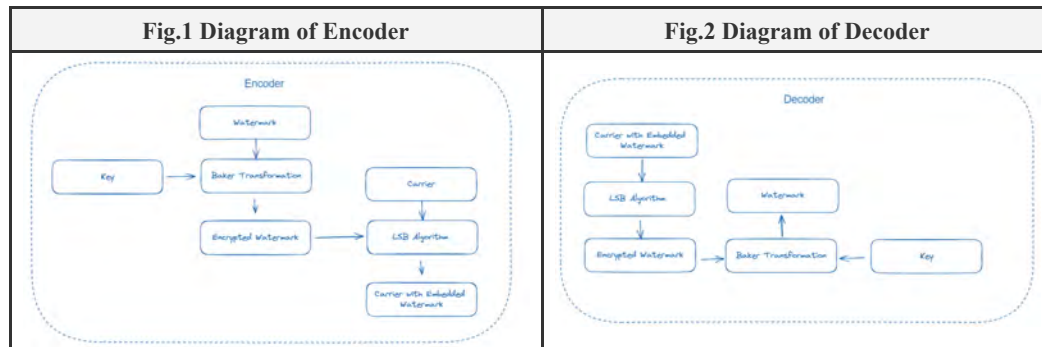
Outline

SUSTech_EE326_Project_Final_Report

1. Introduction
2. Review
3. Method
 - 3.1 Baker Transform
 - 3.1.1 Introduction
 - 3.1.2 Code
 - 3.1.3 Result
 - 3.2 LSB Encoder
 - 3.2.1 Introduction
 - 3.2.2 Result
 - 3.3 LSB Decoder
 - 3.3.1 Introduction
 - 3.3.2 Result
4. Performance
 - 4.1 Indicator
 - 4.2 Robustness Attack Test
5. Summary
6. Reference

2. Review

A complete digital watermarking system is divided into two parts: the encoder and the decoder. Figure 1 shows the structure of the encoder of lsb algorithm, the encoder encrypts the watermark information and then embeds it into the carrier. Figure 2 shows the structure of the decoder of the lsb algorithm. The decoder extracts the watermark from the carrier containing the watermark information, and then decrypts it to obtain the watermark information [1].



The most important parts of digital watermarking technology are encryption technology and embedding technology. After more than two decades of development, researchers have created various cryptographic methods. For example, Baker transform, Arnold transform, FASS curve, phantom square transform, Gray code, life model, etc. [2]. In this project, we use baker transform as the encryption method for watermarking. baker transform is named after the baker's kneading operation: the baker flattens and stretches a square of dough, cuts it in half, and then overlaps it to form a new square of dough.

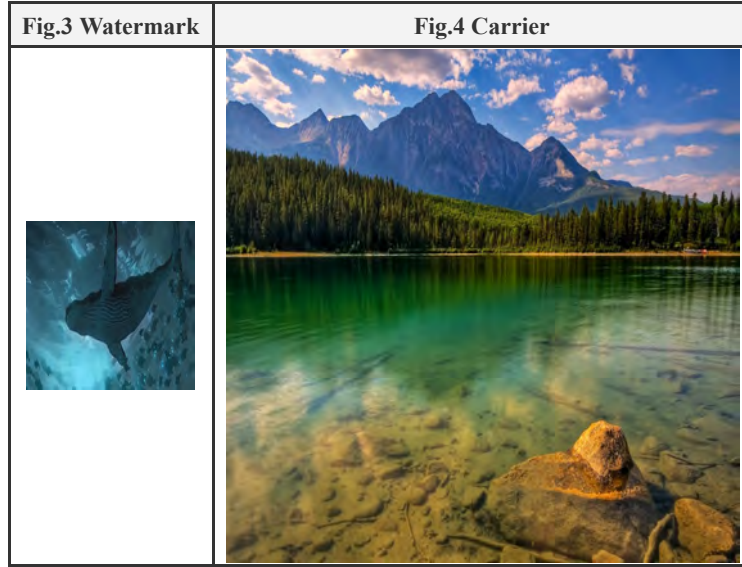
We can embed watermarks in different domains. Early digital watermarking techniques were developed in the spatial domain, and the LSB algorithm is the most typical and well-known digital watermarking technique in the spatial domain. We take a grayscale map as an example, the intensity of each pixel ranges from 0 to 255, so it can be represented by 8 bits. The importance of the information represented by the bits at different positions are different, we call the highest bit the most significant bit (MSB) and the lowest bit the least significant bit (LSB). Since the information represented by LSB is the least important, we can consider using watermarked information to replace the carrier's lsb. This is the idea of the lsb algorithm [4].

<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;"> 1 0 0 1 0 1 0 1 </div>	<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;"> 1 0 0 1 0 1 0 1 </div>
The binary representation of decimal 149, with the LSB highlighted. The LSB represents a value of 1.	The unsigned binary representation of decimal 149, with the MSB highlighted. The MSB represents a value of 128.

But the classical lsb algorithm is less robust and difficult to resist attacks such as noise, compression, smoothing and sharpening. Therefore researchers proposed the method of adding watermark in frequency domain. We convert the carrier to the frequency domain and replace the coefficients of the spectrum with watermark information. The robustness and invisibility of the digital watermarking technique based on frequency domain depend on the frequency band where the watermark is located: when embedded into the high frequency region, the invisibility of the watermark is better and less detectable by the human visual system, but the robustness is poor, especially because the image compression techniques discard the high frequency components and therefore cannot resist compression attacks; when embedded into the low frequency region, the robustness of the watermark is better, but it affects the invisibility. We need to choose the appropriate frequency band according to the usage of the watermark. Besides, we can also add watermarks in the discrete cosine transform domain and wavelet domain.

3. Method

In this project, we use a color image (Figure 3) of size (256,256,3) as the watermark, a color image (Figure 4) of size (1024,1024,3) as the carrier, the baker transform as the encryption method, and the lsb algorithm as the embedding method.



The main work is divided into three parts: baker transform, encoder and decoder.

3.1 Baker Transform

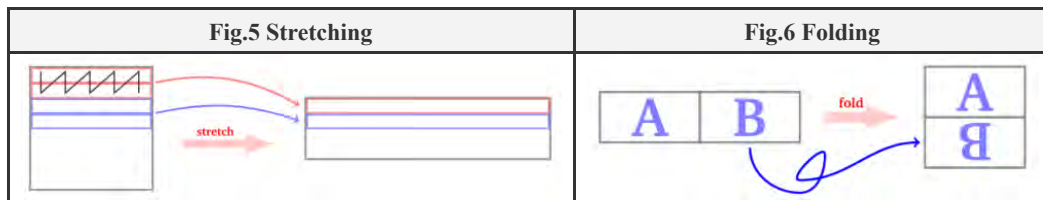
3.1.1 Introduction

The mathematical expression of the baker transform is shown below [3].

$$B(x, y) = \begin{cases} B(2x, \frac{y}{2}), & 0 \leq x < \frac{1}{2} \\ B(2 - 2x, 1 - \frac{y}{2}), & \frac{1}{2} \leq x < 1 \end{cases} \quad (1)$$

This may seem difficult to understand. But we can divide the process of baker transformation into two steps: stretching and folding.

We assume that we have a matrix A of size (n, n) and need to perform the baker transform on the matrix A. First stretch the matrix A so that the matrix A of size (n, n) becomes a rectangular matrix B of size (n/2, 2n). Every two rows (each of length n) produce a single row of length 2n. We mix the values of each row by alternating an upper element and a lower element. Figure 5 illustrates the stretching process [3].



Then we divide the matrix B of size (n/2, 2n) into two halves from the middle, flip the right half up and down, and then move it under the left half to form a matrix C of size (n, n). Figure 6 shows the folding process. The matrix C is the matrix A after one baker transformation [3].

The baker transform has an important feature: a matrix D of size (n, n) is still the matrix D after (2n+1) baker transforms. Based on this feature, the baker transform has become the most popular watermark encryption technique in digital watermarking field. For example, we can perform n times baker transform at the encoder and n+1 times baker transform at the decoder. The number of baker transforms is used as the key. By encrypting the watermark using baker transform, the security of digital watermark can be greatly improved.

3.1.2 Code

- Stretching

```

1 def baker_stretch(array):
2     array = np.asarray(array, dtype=int)
3     m, n = array.shape
4     array_out = np.zeros((int(m / 2), int(n * 2)), dtype=int)
5     for i in range(int(m / 2)):
6         for j in range(int(n * 2)):
7             if j % 2 == 0:
8                 array_out[i, j] = array[2 * i, j // 2]
9             else:
10                 array_out[i, j] = array[2 * i + 1, j // 2]
11     return array_out

```

- Folding

```

1 def baker_fold(array):
2     array = np.asarray(array, dtype=int)
3     m, n = array.shape
4     array_out = np.zeros((int(m * 2), int(n / 2)), dtype=int)
5     for i in range(0, m):
6         for j in range(int(n / 2)):
7             array_out[i, j] = array[i, j]
8     for i in range(m, m * 2):
9         for j in range(int(n / 2)):
10             array_out[i, j] = array[m - i - 1, n - 1 - j]
11     return array_out

```

- Baker Transform

```

1 def baker(array):
2     array_stretch = baker_stretch(array)
3     array_fold = baker_fold(array_stretch)
4     return array_fold

```

3.1.3 Result

We take a 4×4 matrix as an example, and it reverts to the original matrix after 5 baker transformations.

Original Matrix	No.1 Baker Transform	No.2 Baker Transform
<pre> [[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]] </pre>	<pre> iterate No.1 [[1 5 2 6] [9 13 10 14] [16 12 15 11] [8 4 7 3]] </pre>	<pre> iterate No.2 [[1 9 5 13] [16 8 12 4] [3 11 7 15] [14 6 10 2]] </pre>
No.3 Baker Transform	No.4 Baker Transform	No.5 Baker Transform
<pre> iterate No.3 [[1 16 9 8] [3 14 11 6] [2 15 10 7] [4 13 12 5]] </pre>	<pre> iterate No.4 [[1 3 16 14] [2 4 15 13] [5 7 12 10] [6 8 11 9]] </pre>	<pre> iterate No.5 [[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]] </pre>

The example result of baker's transform with a matrix with size (4,4).

3.2 LSB Encoder

3.2.1 Introduction

In general, the encoder encrypts the watermark using the baker transform and then embeds it into the carrier. But the specific steps are a bit complicated. The following describes the steps and code of the encoder.

1. The first step is to read the carrier and watermark. Since this project uses color images as watermarks, we designed the function `reading`, with the path of the image as input and the ndarray of R, G, B channels and image as output.

```
1 | img_src, img_r, img_g, img_b = reading(path=img_src_path)
2 | mark_src, mark_r, mark_g, mark_b = reading(path=mark_src_path)
```

2. The second step is to convert the 2D decimal watermark array of size (256, 256) to a 1D binary watermark array of size (256*256*8). We take the R channel as an example to show the code of this part. This step is repeated for the G,B channels.

```
1 | bin_mark_r = np.zeros((mark_m, mark_n, 8), dtype=int)
2 | for i in range(mark_m):
3 |     for j in range(mark_n):
4 |         bin_mark_r[i, j, :] = dec2bit(mark_r[i, j])
5 | bin_mark_r2 = np.reshape(bin_mark_r, mark_m * mark_n * 8)
```

Since python's binary conversion function `bin()` outputs a string (e.g. `bin(255)='0b11111111'`), I designed the function `dec2bit`, which outputs an 8-bit binary array.

```
1 | def dec2bit(num_dec):
2 |     s = bin(num_dec)[2:]
3 |     while len(s) < 8:
4 |         s = "0" + s # 满足条件的高位补零
5 |     num_bit = np.zeros(8, dtype=int)
6 |     for o in range(8):
7 |         num_bit[o] = s[o]
8 |     return num_bit
```

3. The third step is to pad the 1D binary watermark array of size (256*256*8) to make it of size (1024*1024). This step is not necessary because we don't have to make every pixel of the carrier carry watermark information. But doing so will enhance the ability to fight against clipping attacks.

```
1 | pad_num = img_src_m * img_src_n - mark_m * mark_n * 8
2 | bin_mark_r2 = np.pad(bin_mark_r2, (0, pad_num), mode='wrap')
```

4. The fourth step is to reshape the 1D array of size (1024*1024) into a 2D array of (1024,1024) and then encrypt the watermark information using the baker transform.

```
1 | bin_mark_r2 = np.reshape(bin_mark_r2, (img_src_m, img_src_n))
2 | bin_mark_r2, bin_mark_g2, bin_mark_b2 = enc(bin_mark_r2, bin_mark_g2,
    bin_mark_b2, k=key)
```

5. The fifth step is to replace the carrier lsb. In this project, I replaced the lsb for each pixel point of the carrier image. However, one improvement of the LSB algorithm is to use carrier pixels at specific locations to carry watermark information.

```

1 bin_src_r = np.zeros((img_src_m, img_src_n, 8), dtype=int) # The binary carrier
  array of R channel
2 dec_src_r = np.zeros((img_src_m, img_src_n), dtype=int) # The decimal carrier
  array of R channel
3 for i in range(img_src_m):
4     for j in range(img_src_n):
5         bin_src_r[i, j] = dec2bit(img_r[i, j]) # dec2bin
6         bin_src_r[i, j, 7] = bin_mark_r2[i, j] # Replace the lsb of carrier
7         dec_src_r[i, j] = bit2dec(bin_src_r[i, j, :]) # bin2dec

```

Besides, I also designed the function bit2dec to convert the binary carrier array with embedded watermark information to a decimal array.

```

1 def bit2dec(num_bit):
2     arr = np.asarray(num_bit, dtype=str)
3     s = "0b" + "".join(arr)
4     num_dec = int(s, 2)
5     return num_dec

```

6. The sixth step is to reorganize the carrier of R,G,B channels with embedded watermark information into an image array of size (1024,1024,3) and then save it.

```

1 img_out_rgb = rgb2img(size=(img_src_m, img_src_n), r=dec_src_r, g=dec_src_g,
  b=dec_src_b)

```

3.2.2 Result

In this project, we use Figure 3 as the color watermark and Figure 4 as the carrier. Figure 5 is the complementary watermark after 10 baker transformations. In fact, we binarized it because the baker transformed watermark array is binary and looks completely black if it is not binarized. Figure 6 is the carrier after embedding the watermark and we can not distinguish it from the carrier before embedding at all. Figure 7 shows the difference between the embedded carrier and the original carrier (without binarization), and Figure 8 shows the binarized difference. We observe that the difference is very insignificant and the binarized difference is exactly the encrypted watermark.

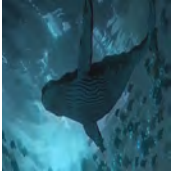
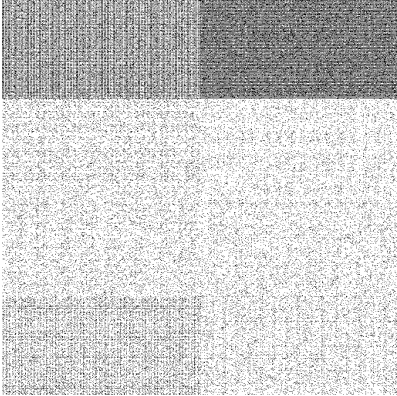


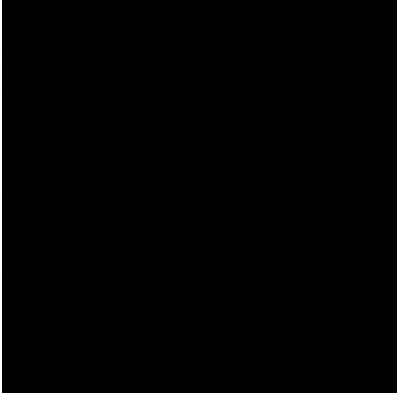
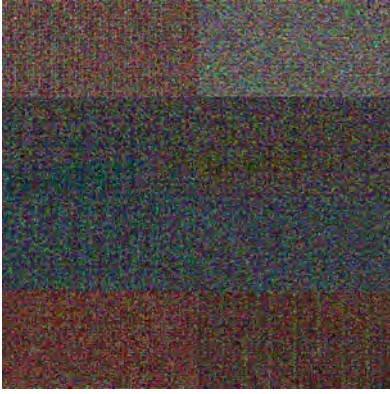
Fig.3 Watermark (256,256,3)	Fig.5 Encrypted Watermark with padding after binarizing(1024,1024,3)	Fig.4 Carrier (1024,1024,3)
		
Fig.6 Embedded Carrier (1024,1024,3)	Fig.7 The Difference between Embedded Carrier and Original Carrier before binarizing (1024,1024,3)	Fig.8 The Difference between Embedded Carrier and Original Carrier after binarizing (1024,1024,3)

Fig.3 Watermark (256,256,3)	Fig.5 Encrypted Watermark with padding after binarizing(1024,1024,3)	Fig.4 Carrier (1024,1024,3)
		

3.3 LSB Decoder

3.3.1 Introduction

The task of the decoder is the opposite of the encoder. The decoder extracts the encrypted watermark from the received image, and then decrypts the watermark according to the key. The steps of decoder are shown below.

1. In the first step, the decoder reads the carrier which carries the watermark information.

```
1 | img_rec, img_rec_r, rec_g, rec_b = reading(path=img_mark_path)
```

2. In the second step, the decoder extracts the watermark from the lsb of the carrier. In this step the decoder first converts the decimal carrier array to binary watermark array, and then extracts the lsb to form a 2D binary watermark array of size (1024,1024).

```
1 | bin_rec_r = np.zeros((rec_m, rec_n, 8), dtype=int)
2 | bin_mark_rec_r = np.zeros((rec_m, rec_n), dtype=int)
3 | for i in range(rec_m):
4 |     for j in range(rec_n):
5 |         bin_rec_r[i, j, :] = dec2bit(img_rec_r[i, j])
6 |         bin_mark_rec_r[i, j] = bin_rec_r[i, j, 7]
```

3. In the third step, the decoder uses baker transform to decrypt the watermark information. The number of baker transforms is determined by the key. Since the image array size we use is (1024,1024), it can be restored to the original array after 21 times of baker transform, and 10 times of baker transform have been performed in the encoder, so 11 times of baker transform are needed in the decoder.

```
1 | bin_mark_rec_r, bin_mark_rec_g, bin_mark_rec_b = enc(bin_mark_rec_r,
    bin_mark_rec_g, bin_mark_rec_b, k=11)
```

4. In the fourth step, the decoder segments the 1D binary watermark array of length (1024*1024), and the length of each segment is (256*256*8). This is because in the encoder we do padding.

```
1 | bin_mark_rec_r = np.reshape(bin_mark_rec_r, rec_m * rec_n)
2 | bin_mark_rec_r1 = bin_mark_rec_r[0: mark_m * mark_n * 8]
```

5. In the fifth step, the decoder converts the 1D binary watermark array of length (256*256*8) to a decimal watermark array of size (256,256, 3).

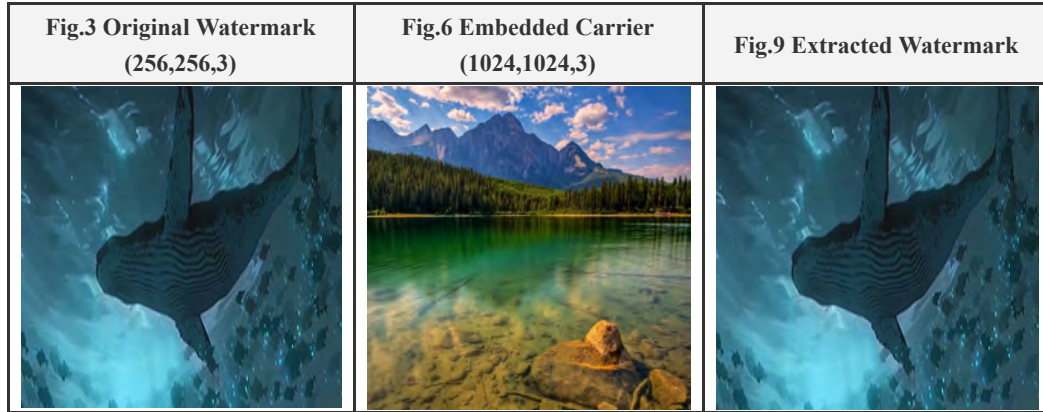
```

1 bin_mark_rec_r1 = np.reshape(bin_mark_rec_r1, (mark_m, mark_n, 8))
2 dec_mark_rec_r = np.zeros((mark_rec_m, mark_rec_n), dtype=int)
3 for i in range(mark_rec_m):
4     for j in range(mark_rec_n):
5         dec_mark_rec_r[i, j] = bit2dec(bin_mark_rec_r1[i, j, :])
6         dec_mark_rec_g[i, j] = bit2dec(bin_mark_rec_g1[i, j, :])
7         dec_mark_rec_b[i, j] = bit2dec(bin_mark_rec_b1[i, j, :])
8 mark_rec_rgb = rgb2img(size=(mark_rec_m, mark_rec_n), r=dec_mark_rec_r,
                             g=dec_mark_rec_g, b=dec_mark_rec_b)

```

3.3.2 Result

Figure 9 shows the final extraction result of the decoder. We can observe that the spatial domain LSB algorithm is a simple and effective digital watermarking technique in the absence of interference.



4. Performance

In part 3, we designed a digital watermarking system based on baker transform and LSB algorithm, and now we need to evaluate the performance of this system. We can evaluate the performance of the system in two ways.

4.1 Indicator

First, there are some indicators (such as PSNR and NC) that can be used to measure the performance of the system.

The Peak Signal-to-Noise Ratio is used to measure the distortion between the embedded watermarked image and the original image. The larger the PSNR value, the less distortion and the better the invisibility of the watermark. The PSNR value ranges from [0,100]. When the PSNR value is greater than 30, the human eye visual system cannot perceive the difference between the watermarked image and the original image.

The definition of PSNR is as below.

$$PSNR = 10 \times \lg \left(\frac{MAX^2}{MSE} \right) \quad (2)$$

$$MSE = \frac{1}{m \times n} \sum_{i=1}^m \sum_{j=1}^n [f(i, j) - g(i, j)]^2$$

where $f(x, y)$ is the original image, $g(x, y)$ is the image with watermark, and MAX is the maximum value of image pixels.

The Normalized Correlation Coefficient (NC) is used to measure the similarity between the original watermark information and the extracted watermark information, and its value range is [0,1].

The larger the NC value, the higher the similarity between the original watermark and the extracted watermark, and the stronger the robustness of the watermarking algorithm; Conversely, the smaller the NC, the lower the similarity between the original watermark and the extracted watermark, and the weaker the robustness of the watermarking algorithm.

The definition of NC is as below.

$$NC = \frac{\sum_{i=1}^m \sum_{j=1}^n (w(i, j) \times w'(i, j))}{\sqrt{\sum_{i=1}^m \sum_{j=1}^n [w(i, j)]^2} \sqrt{\sum_{i=1}^m \sum_{j=1}^n [w'(i, j)]^2}} \quad (3)$$

where w is the original watermark, w' is the extracted watermark, m and n are the number of rows and columns of the watermark image matrix, respectively.

To save the length of the article, I put the code for PSNR and NC in the appendix file.



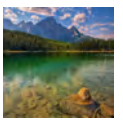








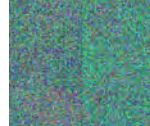
Figure 10 shows the peak signal-to-noise ratio of the carrier after embedding the watermark and the original carrier in R,G,B channels. Figure 11 shows the normalized correlation coefficient between the extracted watermark and the original watermark in R,G,B channels.

Fig.10 PSNR	Fig.11 NC
[51.1365914 51.14051548 51.10452306]	[1. 1. 1.]

We find that the PSNR is 51, which means that the human eye cannot detect the difference between the two images. The NC of the R,G,B channels are all 1, which means that the extracted watermark is exactly the same as the original one. Since we do not add any interference in this test, such a result is a matter of course.

4.2 Robustness Attack Test

We can also test the robustness of this system through various image processing methods. In this project, I tested noise (pretzel noise, Gaussian noise, random noise), histogram equalization, and smoothing filters (mean filter, median filter). The table below shows the test results.

	Pepper noise, probability 0.1	Gaussian noise, mean 10, sigma 255	Random noise, number 100	Histogram equalization	Mean filter, neighborhood size (3,3)	Median filter, neighborhood size (3,3)
Embedded Carrier						
Extracted watermark						

Digital watermarking system needs to consider robustness, security, invisibility, information capacity and other metrics.

We found that this typical LSB digital watermarking system has poor robustness, it only has good resistance to pretzel noise attack and cannot resist other types of attacks. However, since we use the baker transform to encrypt the watermark, there is no need to worry about the security of the system. Since we use a color image of size (m, n) as the watermark and a color image of size (p, q) as the carrier, it is required that $m \times n \geq p \times q$. The information capacity of this system can be increased if we use a binarized image as the watermark or use higher bits to carry the watermark information.

5. Summary

In this project, we design a digital watermarking system based on baker transform and spatial domain LSB algorithm. We encrypt the watermark by Baker transform and embed the watermark into the carrier by LSB algorithm. The spatial domain LSB digital watermarking technique, as an early digital watermarking technique, has a simple idea and poor robustness. Therefore, we can consider the transform domain digital watermarking technique.

Note: Although I try the frequency domain digital watermarking technique, I do not introduce it in the body of the report. This is because I think my code is flawed. The principle of frequency domain watermarking technique is to use watermark information to replace the spectral coefficients of the carrier. However, in practice, I multiplied the watermark array by a factor K and then added it directly to the high frequency region. If the coefficient K is too large (such as $K \geq 25$), the carrier will become dark, and if the coefficient K is too small (such as $K < 10$), the carrier will not change significantly, but the watermark in the frequency domain is very inconspicuous. Perhaps I should use larger coefficient and then perform histogram equalization. But I don't have time to practice this idea.

6. Reference

- [1] 王树梅.(2022). 数字图像水印技术综述. 湖南理工学院学报 (自然科学版)(01),31-36+68. doi:10.16740/j.cnki.cn43-1421/n.2022.01.006.
- [2] 张华熊, 仇佩亮.(2001). 置乱技术在数字水印中的应用. 电路与系统学报 (03),32-36.
- [3] Wikipedia contributors. (2022, January 25). Baker's map. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:11, June 3, 2022, from https://en.wikipedia.org/w/index.php?title=Baker%27s_map&oldid=1067779197
- [4] Wikipedia contributors. (2022, May 18). Bit numbering. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:13, June 3, 2022, from https://en.wikipedia.org/w/index.php?title=Bit_numbering&oldid=1088576658