Image Processing Homework 4

310551089 楊勝皓, L091084 邱楷涵

1 Introduction

In this homework, our purpose is to embed the binary watermark in a color image. First, we discuss the behavior of the watermark.

The watermark is a 290×290 QR-code image. If we scan the QR-code, we will get the information "Hello world". However, outermost area is all white, we only screen the middle 256×256. We use this mark when embedding the watermark. After extracting, we add the outermost area back.

Next, the image we want to watermark is "lena", which is a color image with resolution 1024×1024.

2 Method

2.1 Strategy

Since the resolution ratio of "lena" and watermark is 4, we consider to segment "lena" to 256×256 images with the size 4×4 and hence correspond to the 256×256 watermark. Then , we try to embed information pixel-wisely in the 4×4 sub-image. Note that the the pixel $w = 0$ if the watermark at this pixel is black and 1, otherwise.

However, if we use original watermark to embed the image, the watermarked image may emerge some pattern. Therefore, we do the permutation on the watermark to avoid that problem. When decoding, we also need this permutation to get the original watermark back.

Next, since the image is a color image, we hide the information in each r, g, b channel. Hence, we can extract three watermarks. Since the watermark is binary image, we determine final watermark by voting. For each pixel, if any two channel have same values at that pixel, then we assign that value to the final watermark.

2.2 Embedding

For the embedding, the algorithm is showing as below:

1. Split the color into r, g, b channel.
2. Shuffle the watermark. Note that three channels have different permutation list which can prevent the local attack.
3. In each channel, we segment original image into 256×256 sub-image with size 4×4 which corresponding to the shuffle watermark.
4. For each sub-image M ,which is 4×4 matrix, and corresponding watermark pixel w:
   a) Do the DCT on M and get another matrix D.
   b) Do the following change:

$$D_{11} = \begin{cases} D_{11} + C & \text{,if } w = 0. \\ D_{11} - C & \text{,if } w = 1. \end{cases}$$

   Here $C = 19$ is a constant.
   c) Do inverse DCT and get the new sub-image $\tilde{M}$.
5. Merge all the sub-image $\tilde{M}$, clip off the value outside the range $[0, 255]$ and turn it into uint8 type.
6. Merge three channels and get the watermarked image.

We apply this method on the image and get PSNR value equal to 34.71. The following is our result:

2.3 Extracting

For the extracting, similar we segment the image:

1. Split the color into r, g, b channel.
2. In each channel, we segment watermarked image into 256×256 sub-image with size 4×4.
3. For each sub-image $\tilde{M}$:

a) Do the DCT on $\tilde{M}$ and get another matrix $\hat{D}$.

b) Here we need D defined in embedding method, and the extracted pixel is defined as

$$\hat{w} = \begin{cases} 0 & \text{,if } D_{11} \leq \tilde{D}_{11}. \\ 1 & \text{,if } D_{11} > \tilde{D}_{11}. \end{cases}$$

4. Permute the output back we can get three watermarks. (For attack cases, we will apply some morphological methods, mentioned in next section, to enhance the watermark.)

5. Eventually, voting three watermarks and add the outermost back we can get final watermark.

The bit error rate for our recovery is 0.

2.4 Morphological processing

If there is no any transformation on the watermarked image, the watermark can be recovered completely. However, sometimes the image may be transformed and hence they would have some noise on the extracted watermark. In the next section, fewer transformation would be considered. Hence, we provided some method.

Since the QR-code has significant pattern, so we just need to remove the outside noise and fill up the inside noise. Here we provided a simple method.

- First, use 9×9 Gaussian blur to the watermark and set a threshold $T = 178/255$. The formula is given as:

$$\hat{w} = \begin{cases} 0 & \text{,if } \hat{w} < \frac{178}{255} \\ 1 & \text{,if } \hat{w} \geq \frac{178}{255} \end{cases}$$

- Next, use the binary closing operator by 5×5 kernel whose all entries equal to 1. This operator can fill up the missing pattern but enlarge the outside noise.

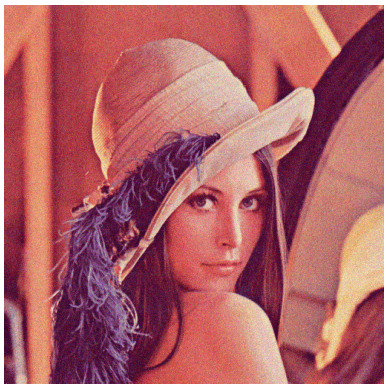
- Finally, use the binary opening operator by 3×3 kernel whose all entries equal to 1. This operator can remove the outside black noise and get better image.

3 Attack

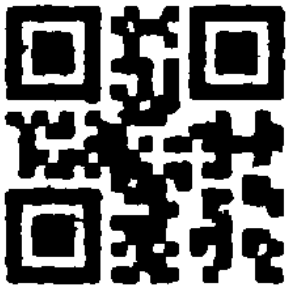
In this section, we will show our recovered watermarks from several common attacks.

3.1 Gaussian noise

The marked image after attacked by Gaussian noise has the result:



,and the extracted watermark is



with bit error rate is 0.068. However, it cannot be scanned.

3.2 Gaussian blur

The marked image after attacked by Gaussian blur has the result:

,and the extracted watermark is

with bit error rate is 0.020. It can be scanned and got information.

3.3 Compression

The marked image after attacked by compression has the result:

,and the extracted watermark is

with bit error rate is 0.048. It can be scanned and got information.