浙江大学

课程名称：	多媒体安全
姓　　名：	庞懿非
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
学　　号：	3190104534
指导教师：	黄劲
完成时间：	2022 年 6 月 17 日

实验三：E_BLK_8/D_BLK_8 系统测试

一、实验目的

1. 了解 E_BLK_8/D_BLK_8 系统的基本原理。

2. 了解 Hamming Code 和 Trellis Code 的工作原理。

3. 掌握 Correlation Coefficient 的计算。

二、实验内容与要求

1. 实现基于 E_SIMPLE_8/D_SIMPLE_8 系统的 E_BLK_8/D_BLK_8 系统。要求使用 Correlation Coefficient 作为检测值。

2. 设计一张水印，选择嵌入强度 $\alpha = \sqrt{8}$，使用该水印测试基于 E_SIMPLE_8/D_SIMPLE_8 系统的 E_BLK_8/D_BLK_8 系统应用于不同封面时的检测准确率。要求封面数量不少于 40 张。

3. 实现基于 Hamming Code 或 Trellis Code 的 E_BLK_8/D_BLK_8 系统。

4. 使用固定的水印和固定的嵌入强度，测试基于 Hamming Code 或 Trellis Code 的 E_BLK_8/D_BLK_8 系统应用于不同封面时的检测准确率。这里 α 取值根据所采用的 Hamming Code 或 Trellis Code 编码方式选定。比较在信息末尾添加两个 0 比特是否有助于提高检测的准确率，如果可以，请解释原因。

5. 比较基于不同系统，E_SIMPLE_8/D_SIMPLE_8 和（基于 Hamming Code 或 Trellis Code 的）E_BLK_8/D_BLK_8 系统的检测准确率，试分析原因。

三、实验环境

1. IDE: MATLAB, Eclipse（Java）

2. OS: Windows 10

四、实验过程

0.背景与思路介绍

实验三在实验二的基础上，使用了 **media space** 到 **marking space** 的映射，缩小了空间维度，但是在计算上更加简单。整个实验分为两大部分，第一个是原始的 E_BLK_8/D_BLK_8 系统，另一种是使用了 ECC 的 E_BLK_8/D_BLK_8 系统，在实验中我使用的是 **hamming code**。两个系统都可以分为：设计水印、获得标记空间（block 平均值），加入水印到标记空间，从标记空间映射到全空间（并生成图片），检测信息，reencode 判断信息的 **Correlation Coefficient**，最后判断信息是否正确（第二个系统要使用 hamming code 进行校验）。

1.设计水印

以为图片的每个块（block）的大小 8*8，使用高斯分布函数一次性生成等大小的 8 个水印，并且将 8 个水印的数值按照字符串存储在 hex 文件中。由于高斯分布（正态分布）生成的数值为随机浮点数，小数点后位数可能过长，所以我统一使用 6 位小数存储在文件中作为水印，之后直接从文件中读取浮点数作为水印信息。

```
Random r = new Random();
(int) (r.nextGaussian() * 1);
```

此外，我们还需要对生成的水印进行标准化和相关性检测，由于我们的水印是在 64 维空间生成的，远远小于之前常用的 512*512 维空间，所以多组水印之间的线性无关性并不是随意生成就可以满足，相反，这些水印之间往往高度相关。

```
public static double wrLCCheck(int size, int length, String filename){
    //...
    for(int i = 0; i < length; i++) {
            for(int j = i + 1; j < length; j++) {
                for(int z = 0; z < 64; z++) {
                    linear_c += wr_data_2d[i][z] * wr_data_2d[j][z];
                }
                System.out.println("--" + i + " and " + j + " linear_c = " + linear_c);
                if(linear_max < linear_c) {
                    linear_max = linear_c;
                }
                if(linear_min > linear_c) {
                    linear_min = linear_c;
                }
            }
        }
        System.out.println("max co is " + linear_max/64);
        System.out.println("min co is " + linear_min/64);
    //return bigger one
}
```

于是，我设计了如上的函数，在对 8 个水印都标准化之后（均值为 0，方差为 1），检测所有水印之间的线性相关性。并且要保证相关性的绝对值的最大值小于 0.3。这里非常重要的一点就是我们一组水印是使用水印的正负两个值，如果两组水印高度负相关，其实就等价于组水印高度正相关，所以正负对于我们来说都是同样重要的。在循环进行生成之后，我们获得了理想的结果。

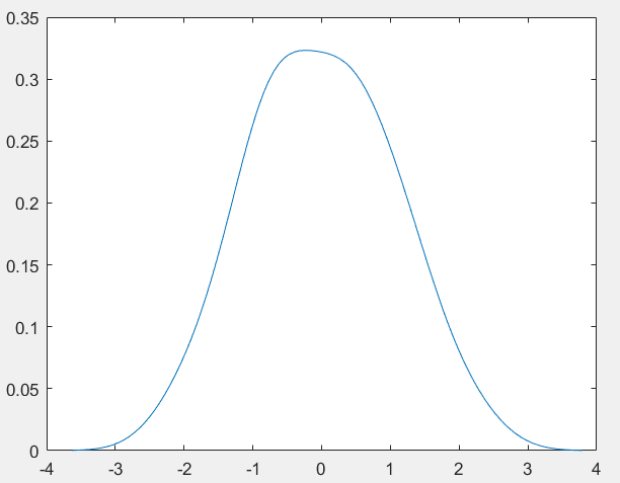
```
while(true) {
    getRandomint(size * length,filename);
    double wr_data[] = new double[size * length];
    readFromFile(filename,wr_data);
    Normalize(size,length,wr_data,filename);
    double ret = wrLCCheck(size, length, filename);
    if(ret < 0.3) {
        break;
    }
}
```

在实际情况中，嵌入端和检测端不共享文件，通常是通过一个共享的种子（即密钥）作为随机数生成器然后生成出相同的水印编解码。在这里我们统一利用了一个共享的水印文件，可以方便查看水印的分布情况（将水印可视化），在利用水印信息的时候也更加便捷。

将水印可视化的过程如下所示：

```
dat1=load('wr1.hex');
x=dat1(:);
x=reshape(x,8,512*512);
% tabulate(x(2,:));
x1 = sort(x(1,:));
[f,xi]=ksdensity(x1);
plot(xi,f);
[mu12,sigma12] = normfit(x1);
```

将某个水印文件加载并且转换为 8*64 的二维矩阵，之后对其中某一个水印绘制曲线分析；该水印文件的第 1 个水印的概率密度曲线如下所示：



如图所示，该 8*8 水印几乎良好的符合高斯分布曲线，使用正态分布的估计，均值约为 $-e^{-08}$，标准差为 1.008，符合要求。

两个系统中生成的水印的线性相关性检测结果为（左边为第一个系统，8 组水印；右边为加入校验的系统，hamming code 对 8bit 信息有 4 位校验码，所以一共是 12bit，也就是生成了 12 组水印）：

```
max co is 0.19518659786354722
min co is -0.20991613537903114
```

```
max co is 0.29613026226945327
min co is -0.2602381112014996
```

2. 获得标记空间

由于我们实验是在标记空间上进行的，我们需要将图片从全空间投影到标记空间，实验中使用了求 8*8 块平均值的简单方式进行。

```java
public static void getAverageBlock(int [][]pixels, double [][]average_block) {
    int length = pixels.length;
    int width = pixels[0].length;
    for(int i = 0; i < length ; i++) {
        for(int j = 0; j < width; j++) {
            average_block[i%8][j%8] += pixels[i][j];
        }
    }
    int block_number = length * width / 64;
    for(int k = 0; k < 64; k++) {
        average_block[k/8][k%8] /= block_number;
    }
}
```

将图片分成多个 8*8 的块，然后将块的每一个位置的像素值求平均得到了标记空间。

3. 加入水印到标记空间

在嵌入水印时我们统一 8*8 的块，与水印信息相匹配。为了避免可能的问题，我们使用长宽分辨率都为 8 的倍数的图片来进行处理。

不难发现，在 E_BLK_8/D_BLK_8 中，嵌入水印信息仍然是使用了 E_BLIND 的方法，唯一的区别就是，本系统中一次性传输 8bit 信息，要对 8 个水印进行混合。由于每个水印的嵌入都是使用公式 $c_w = c_o + \alpha c_m$，即使用加法将水印加入到原图中，所以这里我们先根据要传输的信息，将所有的水印提前进行合成，之后生成一个新的水印嵌入到图片中。

水印合成的部分很简单。如果传输的 bit 为 1，则加该水印，为 0 则减该水印；最后将得到的水印结果除以一个标准差，使得总水印的方差为 1。

```java
public static void getCompWr(int []message, int length, double []ori_wrs,double []result){
    //according to the message
    //calculate the compound watermarks to result
    //calculate the mean of the result and the variance of the result
    //devide the standard variance, make the new watermark have unit variance
}
```

嵌入水印的过程很简单，使用标记空间 + $\sqrt{8}$ * 水印值（使用$\sqrt{8}$是因为在前面将合成水印方差变为 1 的过程中，合成水印基本由正交的 8 个单位方差的向量组成，总方差大约为 8，除以标准差之后每个水印的值都相对上只是原来的$1/\sqrt{8}$，这里将 α 的值取作$\sqrt{8}$，等价于将每个水印的强度恢复到 1，即和实验一中 E_BLIND 具有近乎相同的水印嵌入强度）。在完成这一步后，我并没有立刻将像素值进行取整和限定范围等，而是检查了它的有效性，如下图所示：

```
getAverageBlock(pixel_data,average_block);
/* get watermark --- result */

double [][]new_average_block = new double [8][8];
adding_wr(average_block,result,new_average_block,length);

/* calculate the Zcc */
String outputpath = "zcc_m255_hamming_effective.txt";
Zcc.allZcc(new_average_block, wr, length, imagePath, outputpath, result);
```

最后，相应的 effectiveness 计算结果也存储在了对应文件中。

4. 从标记空间映射到全空间

获得了加入水印的标记空间后，就需要将水印恢复到全空间中。在实验中，需要将每个标记空间像素的变化值加到图片的每个块中对应位置的像素上。效果其实等价于对图片的每个块都加入 8*8 的水印值。

当然在这一过程中，如果新的像素值不在［0,255］的区间，我们要把值 snapping 到 0 或者 255 的有效范围内，最后的结果要四舍五入到整数生成图片。

```
    public static void restoreBlockToPixels(int [][]pixels, double [][]average_block, int [][]new_pixels,
double [][]new_average_block) {
        double [][]difference = new double [8][8];
        int size = 64;
        for(int i = 0; i < size ; i++) {
            difference[i/8][i%8] = new_average_block[i/8][i%8] - average_block[i/8][i%8];
        }
        int length = pixels.length;
        int width = pixels[0].length;
        for(int i = 0; i < length ; i++) {
            for(int j = 0; j < width; j++) {
                new_pixels[i][j] = (int) Math.round((pixels[i][j] + difference[i%8][j%8]));
                if( new_pixels[i][j] > 255) {
                    new_pixels[i][j] = 255;
                }else if(new_pixels[i][j] < 0) {
                    new_pixels[i][j] = 0;
                }else {
                    /* do nothing */
                }
            }
        }

    }
```

在使用生成的一个水印（使用文件“a.hex”中存储的水印）对多张图片嵌入的时候，我使用了遍历文件夹的方式，对“data”文件夹中的所有图片分别加入了 0，255 以及 101（01100101）的信息，生成了 120 张图片存储到了对应的文件夹中。（同实验二）

其余方法，包括读取图片信息，使用新的像素值生成文件，将水印嵌入图片等，和实验一完全一致，这里不予赘述。

5. 计算相关系数 Correlation Coefficient

不同于线性相关性的计算，这里相关系数的结算结果是两个向量之间的夹角的余弦

值。

$$z_{cc}(\mathbf{v}, \mathbf{w}_r) = \frac{\tilde{\mathbf{v}} \cdot \tilde{\mathbf{w}}_r}{\sqrt{(\tilde{\mathbf{v}} \cdot \tilde{\mathbf{v}})(\tilde{\mathbf{w}}_r \cdot \tilde{\mathbf{w}}_r)}},$$

z_{cc} 计算之前先要将两个向量的均值调整为 0，最后计算点积，再除以两个向量的模长。

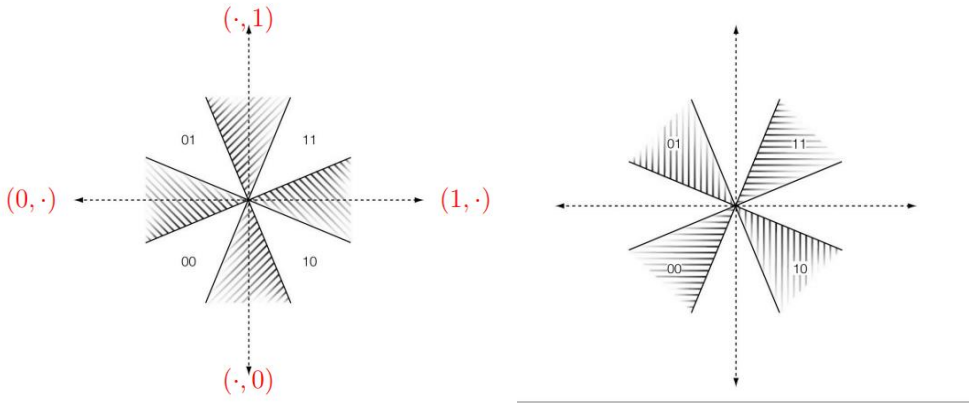
具体实现上，只需要将图片的标记空间标准化（均值为 0，方差为 1），就可以和对应的水印信息计算点积，由于水印本身就是标准化的，计算结果符合 Correlation Coefficient 的定义。

```
public static double testZcc(double [][]block_data, double []wr) {
    double [][]new_block_data = new double[8][8];
    normalize(block_data, new_block_data);
    return calculation(new_block_data,wr);
}
```

Correlation Coefficient 本质上还是进行点积，但是通过提前变均值为 0 以及除以向量模长的方法，可以很好的反映两个向量的夹角等信息，这个相关系数也比线性相关性有更好的性质。

6. reencode 判断信息

Reencode 在使用 Correlation Coefficient 的系统中非常有效，因为在多组水印加入之后，图片与每组水印向量的夹角并不小，也就是对单一水印的相关系数并不大，而是对整个合成水印有较大的相关系数，如下图：



所以我们在实验中先用 z_{lc} 的正负值判断每组水印对应的信息是 0 还是 1，再将信息 reencode 得到合成水印，最后计算标记空间和合成水印的相关系数 Correlation Coefficient，即可以得到很好的结果，具体算法如下所示：

```
double [][]average_block = new double [8][8];
add_wr.getAverageBlock(pixel_data,average_block);
for(int i = 0; i < length; i++) {
    /* calculate the corelation */
    double result = calculation(average_block, wr_data_2d[i]);
    if(result > 0) {
        de_message[i] = 1;
    }else {
        de_message[i] = 0;
    }
}
double [] result_tmp = new double [64];
add_wr.getCompWr(de_message,length,wr_data,result_tmp);
double cc = testZcc(average_block,result_tmp);
```

7. hamming code 校验

8bit 信息的汉明码需要 4 个校验位，如下所示，R 为校验位，D 为数据位，要使得每个校验位的校验序列中 1 的个数为偶数（偶校验）。汉明码的理解方式有很多，比如下面 12 位中，R1 校验的序列为所有的奇数编号序列，这个序列中 1 的个数为偶数。

海明码中的位置	1	2	3	4	5	6	7	8	9	10	11	12
所在位的标识	R1	R2	D3	R4	D5	D6	D7	R8	D9	D10	D11	D12
参与R1校验	X		X		X		X		X		X	
参与R2校验		X	X			X	X			X	X	
参与R4校验				X	X	X	X					X
参与R8校验								X	X	X	X	X

以 101 为例，8 位信息为 01100101，从左到右填入 D 区域，这时根据上面的计算方法，可以得到结果为：1001 1100 0101。这样的汉明码可以检错 1 位，纠错两位。检错即是将 4 个校验位重新计算，如果出错就记录下来，最后将出错的检测位的标号加起来就可以得到错误的位置。比如，传输的信息变成 1001 1100 0111，那么 R1，R2，R8 出错，对应的也就是 D11 传输错误。

8. 阈值设定和性能统计（数据分析）

两个系统都通过对 40 张原图以及 120 张生成图片的 Correlation Coefficient 绘制图像，计算 False Positive/Negative Rate 和解码的准确率。在第二个系统中，额外进行了汉明码校验。

五、实验分析与结论

系统 1

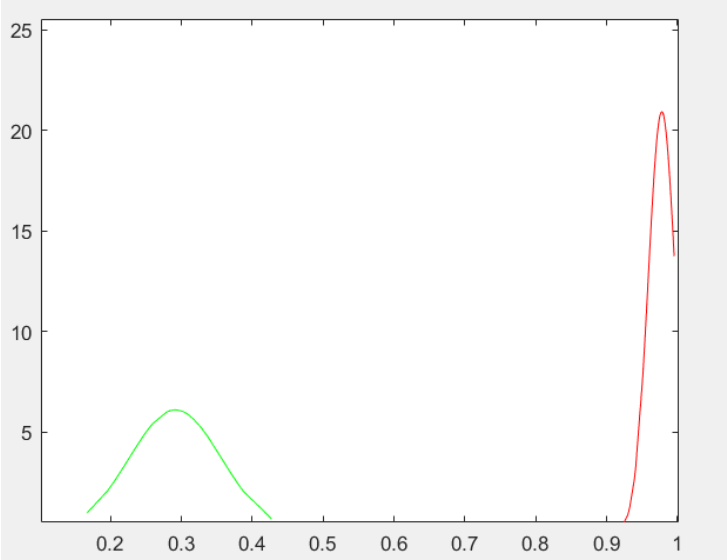
水印的生成与水印的性质检验在上一章节已经详细介绍过了，在这里我使用生成的水印按照传输信息 0，101，255 生成了 120 张图片，如下所示：

4 张图片几乎完全没有视觉上的差异，分别为加入水印信息 0 的 Lena 图（左上），加入水印信息 101 的 Lena 图（右上），加入水印信息 255 的 Lena 图（左下），未修改的 Lena 图（右下）。4 张图与水印的线性相关性检测值如下所示：

图片\水印\z_{cc}	Wr_compound
lena512.bmp	0.38716
wrm0_lena512.bmp	0.99326
wrm255_lena512.bmp	0.98867
wrm101_lena512.bmp	0.99024

明显可以看出，加入水印信息后的图片标记空间和对应水印高度相关，尽管这些图片看起来完全没有差别。类似的，其他图片也都按照加入的水印信息存储在相应的文件夹中，其余图片的相关数据也都存储在“**record.xlsx**”文件中（具体请看文档 Readme）。

在第一部分，我对 160 张图片（120 张生成的加水印图与 40 张原图）的相关系数分别进行统计和正态分布模拟，得到如下结果图：



由于我选择了非常良好的水印，有水印和无水印的相关系数分布有着明显的差异

一般情况下，我们认为需要取一个阈值 τ_{lc} 判断水印的存在性。

结合概率分布曲线，我选择 0.8 作为阈值 τ_{lc}。即，水印的 Correlation Coefficient 高于 0.8 则认为是有水印信息，其他情况认为无水印。

水印图片的检测定义：reencode 后检测到的 Correlation Coefficient 超过了阈值。

水印信息的定义：水印与图片的线性相关性检测值为正，则认为该 bit 为含有信息 1，否则认为该 bit 位含有信息 0。

False Positive Rate:

以 0.8 作为 Correlation Coefficient 的阈值，40 张原始图片都被认为没有添加水印，即假阳性率为 **0%**。

假阳性的含义是：没有水印的图片被检测出有水印，如此可见在水印性质良好的情况下，这个概率非常小。

False Negative Rate:

利用“record.xlsx”的 P 列的统计数据，我们发现在 120 张嵌入水印的图片中，所有的图片都被检测出有水印，假阴性率也为 **0%**。这也是由于我们选择了性质良好的多组水印，水印之间几乎没有干扰，整体的效果也非常好。

解码的准确率：

解码的信息通过如下公式在 Excel 中计算：

```
= (B4 > 0) * 128 + (C4 > 0) * 64 + (D4 > 0) * 32 + (E4 > 0) * 16 + (F4 > 0) * 8
+ (G4 > 0) * 4 + (H4 > 0) * 2 + (I4 > 0) * 1
```

计算结果为根据检测信息的定义计算的信息值。（统计结果在“record.xlsx”的 Q 列）

在所有被检测出有水印的图片中，解码的信息和编码的信息完全一致。最终解码的准确率（正确解码的概率）为 **100%**。未加水印的图片的信息值则千变万化，不过它们都不被认为含有水印。

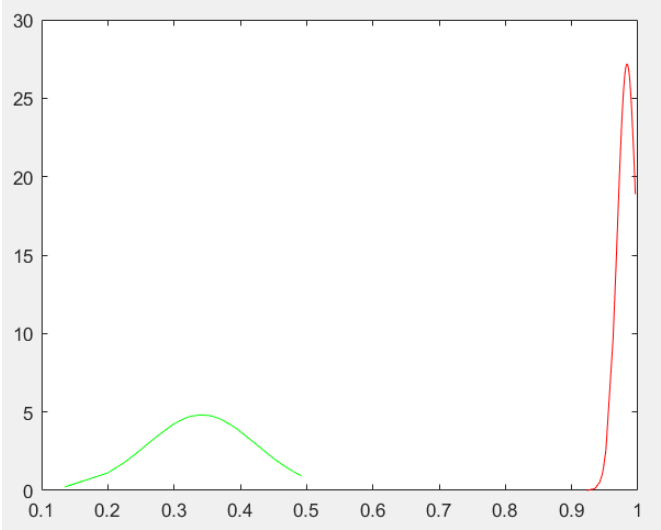
系统 2

系统 2 和系统 1 几乎完全类似，只是在加入信息前和检测信息后进行了校验，所以我们也对应生成了 12 组水印，对应 8 个信息位和 4 个汉明码校验位。

4 张图片几乎完全没有视觉上的差异，分别为加入水印信息 0 的 Lena 图（左上），加入水印信息 101 的 Lena 图（右上），加入水印信息 255 的 Lena 图（左下），未修改的 Lena 图（右下）。（4 张图的水印信息都有 4 个汉明码校验位）4 张图与水印的线性相关性检测值如下所示：

图片\水印\z_{cc}	Wr_compound
lena512.bmp	0.30401
wrm0_lena512.bmp	0.99331
wrm255_lena512.bmp	0.9927
wrm101_lena512.bmp	0.99422

将相关系数分别进行统计和正态分布模拟，得到如下结果图：



结合概率分布曲线，我选择 0.8 作为阈值 τ_{lc}。即，水印的 Correlation Coefficient 高于 0.8 则认为是有水印信息，其他情况认为无水印。

False Positive Rate:

与系统 1 的实验类似，这里我们同样相同的公式进行判断。

以 0.8 作为 Correlation Coefficient 的阈值，40 张原始图片都被认为没有添加水印，即假阳性率为 **0%**。

False Negative Rate:

利用“record.xlsx”的 P 列的统计数据，我们发现在 120 张嵌入水印的图片中，所有的图片都被检测出有水印，假阴性率也为 **0%**。这也是由于我们选择了性质良好的多组水印，水印之间几乎没有干扰，整体的效果也非常好。

解码的准确率：

解码端，我们对检测出有水印的图片进行 ECC（此处使用汉明码）校验，对 R1,R2,R4,R8 分别使用公式：

```
=XOR(D221>0,F221>0,H221>0,J221>0,L221>0)=(B221>0)
=XOR(D221>0,G221>0,H221>0,L221>0,K221>0)=(C221>0)
=XOR(F221>0,H221>0,G221>0,M221>0)=(E221>0)
=XOR(J221>0,M221>0,L221>0,K221>0)=(I221>0)
```

ECC 校验结果全部正确，可从“record.xlsx”的 R,S,T,U 列的统计数据得出。

综上，在所有被检测出有水印的图片中，解码的信息和编码的信息完全一致。最终解码的准确率也为 **100%**。

思考题

比较在信息末尾添加两个 0 比特是否有助于提高检测的准确率，如果可以，请解释原因：

在信息末尾增加两个 0，即信息的末两位的信息是固定的，如果最终的信息不满足这样的要求，那么我们认为信息有误（概率很低）或者不含水印。这个能够降低系统的 false positive rate，使得一些侥幸被检测出有水印的未加水印的图片因为末两位信息不为 0 而被判断为不含水印信息。

当然，加入确定的信息也可以帮助我们检查信息是否正确，有时候图片被大面积修改，然后校验码仍然匹配，这时候可以帮助我们识别一些可能的传输错误。（有可能是短时间信道被污染）

比较基于不同系统，E_SIMPLE_8/D_SIMPLE_8 和（基于 Hamming Code 或 Trellis Code 的）E_BLK_8/D_BLK_8 系统的检测准确率，试分析原因。

从整体层面来看，基于 ECC 的 E_BLK_8/D_BLK_8 的检测准确率（包括 false positive rate， false negative rate，解码的准确率等）都要优于 E_SIMPLE_8/D_SIMPLE_8 系统。

尽管在 E_SIMPLE_8/D_SIMPLE_8 系统中，水印的维度非常高，每组水印之间的线性相关系数很小，但是由于在实验中仅仅使用 D_LC 进行相关性检测，具有相对上较高的 false positive rate，有些情况下一些加入了水印的图片与水印的线性相关性也不高，相对上容易出现 false negative rate。在解码的准确率方面，该系统整体上效果还可以，但是相比于带有 ECC 的 E_BLK_8/D_BLK_8 系统来说还是相差了很多倍。

在 E_BLK_8/D_BLK_8 系统中，尽管水印的维度只有 64 维，但是在多次生成之后还是可

以得到相关性不大的多组水印。在相关性检测上使用了 coefficient correlation 代替了线性相关性，并且使用 reencode 的方式计算相关性，很大程度上保证了检测的准确性，降低了 false positive rate 和 false negative rate。从前面的实验结果也可以看到，没加水印和加入水印的图片之间的差异是显著的。并且，该系统使用了 ECC 校验码，能够对识别的比特流进行 1 位的纠错，使得该系统具有更高的解码准确率。

六、实验感想

这个实验基于实验二进一步设计，但是加入了很多新的元素，包括 ECC，水印性质测试，reencode 等等。在水印的存储上，由于水印的维度大幅减少，文件的读取速度变得异常快，在实验二中出现的速度慢的问题这里完全消失。

这次的实验过程中，识别信息处我只是简单的进行线性相关性的计算，这里主要的关注点是正负。在 reencode 的地方我使用了 **coefficient correlation**，使用的 Z_{CC} 的计算方式得到两组向量的余弦值（夹角大小）。

由于之前想先复习完再完成实验，但考试前两天突然生病，导致实验进度受阻，现在才写完 lab3 的报告。