实验3：E_BLK_8/D_BLK_8 系统测试

课程名称： 信息隐藏与数字水印技术

实验项目名称： E_BLK_8/D_BLK_8 系统测试

学生姓名： 黄炯睿 学号： 3170103455

电子邮件地址： jiongrui_huang@zju.edu.cn

实验日期： 2020 年 6 月 11 日

实验目的

1. 理解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。

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 α = √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HammingCode或TrellisCode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。

实验环境

本次实验使用MATLAB2017进行，所有脚本文件均在附件的压缩文件之中

实验过程

实现基于 E_SIMPLE_8/D_SIMPLE_8 系统的 E_BLK_8/D_BLK_8 系统

- 基于E_SIMPLE_8的E_BLK_8

```
function [wimg] = Simple_E_BLK_8(img,message, wi, alpha)
[width, height] = size(img);
img = im2double(img);
w = (zeros(size(wi{1}))); % wi里是八个8*8矩阵
for i=1:size(wi,2)
    if(message(i)==1)
       w = w + wi{i};
    elseif(message(i)==0)
        w = w - wi{i};
```

```
    end
end
w_mean = mean(mean(w));
w = w - w_mean;
w_std = std2(w);
w = w / w_std;

% extract mark from img before modification
v = zeros(8,8);
nvo = zeros(8,8);
n = zeros(8,8);
for i=1:width
    for j=1:height
        imod8 = int32(mod(i-1,8))+1;
        jmod8 = int32(mod(j-1,8))+1;
        nvo(imod8, jmod8) = nvo(imod8, jmod8) + double(img(i,j));
        n(imod8, jmod8) = n(imod8, jmod8) + 1;
    end
end
v = nvo./n;

% use blind embedding to choose a new vector in mark space
% that is close to a given extracted mark and (hopefully) inside
% the detection region
vw = zeros(8,8);
vw = im2double(v) + alpha*w;

% modify an image so that when entered into ExtractMark
% it will produce (approximately) a given mark
delta = n.*vw -nvo;
for i=1:width
    for j=1:height
        imod8 = (mod(i-1,8))+1;
        jmod8 = (mod(j-1,8))+1;
        oldPixel = double(img(i,j));
        newPixel = double(img(i,j))+delta(imod8,jmod8)./n(imod8,jmod8);
        if(newPixel <0)
            img(i,j) = 0;
        elseif(newPixel>255)
            img(i,j) = 255;
        else
            img(i,j) = newPixel;
        end
        n(imod8,jmod8) = n(imod8,jmod8) - 1;
        delta(imod8,jmod8) = delta(imod8,jmod8) - (img(i,j)-oldPixel);
    end
end
wimg = img;
end
```

`img` 是要嵌入进的cover image，数据类型是uint8。

`message` 是要嵌入的信息，是一个一行n列的logical矩阵，表示要嵌入信息的二进制，其中n是用户可以控制的值。

`wi` 是一个元胞数组，存放了n个高斯分布的矩阵(水印)，矩阵大小为8*8。

`alpha` 是嵌入强度，由用户提供，本次实验中均为$\sqrt{8}$。

`wimg` 是返回值，表示嵌入信息后的带水印的图片。

- 基于D_SIMPLE_8 的D_BLK_8

```
function [wm, cc] = Simple_D_BLK_8(wimg, wi, threshold)
[width, height] = size(wimg);
wimg = im2double(wimg);
wm = [];

% Extract Mark
nvo = zeros(8,8);
n = zeros(8,8);
for i=1:width
    for j=1:height
        imod8 = int32(mod(i-1,8))+1;
        jmod8 = int32(mod(j-1,8))+1;
        nvo(imod8, jmod8) = nvo(imod8, jmod8) + double(wimg(i,j));
        n(imod8, jmod8) = n(imod8, jmod8) + 1;
    end
end
v = nvo./n;

% decode
for i=1:size(wi,2)
    tmp = wi{i};
    product = sum(sum(im2double(v).*tmp));
    wm(i) = double(product/(8*8));
end
wm = wm >0;

% re-encoding
w = (zeros(size(wi{1})));
for i=1:size(wi,2)
    if(wm(i)==1)
       w = w + wi{i};
    elseif(wm(i)==0)
        w = w - wi{i};
    end
end

% Correlation Coefficient
v_mean = mean(mean(v));
vw = sum(sum((v-v_mean).*(w-mean(mean(w)))));
vv = sum(sum((v-v_mean).*(v-v_mean)));
ww = sum(sum((w-mean(mean(w))).*(w-mean(mean(w)))));
if( abs(vv * ww) < 0.0001 )
    cc = 0;
else
    cc = vw / sqrt( vv * ww );
end

if((cc) < threshold)
    wm = [-1,-1,-1,-1,-1,-1,-1,-1];
end
end
```

`wimg` 是一张可能带有水印的图片。

`wi` 是一个元胞数组，存放了n个高斯分布的矩阵(水印)，矩阵大小为8*8。

`threshold`是通过检测值是否超过阈值来判断该图片是否含有水印。

`wm` 是解码产生的信息，如果该图片没有水印，则输出为[-1,-1,-1,-1,-1,-1,-1,-1]。

`cc` 是检测值，当低于阈值时会认为图片中没有水印信息。

- Generate Watermark

```
function [wi] = GenerateWatermark(num,x,y,seed)
    wi = cell(1,num);
    if nargin == 3   % 如果参数里没有seed
        for i=1:num
            wi{1,i} = (randn([x,y]));  %正太分布，均值为0  方差为1
        end
    elseif nargin == 4  % 如果参数里有seed
        for i=1:num
            randn('seed',seed + i); % 可以保证同一个seed生成的水印都一样
            wi{1,i} = (randn([x,y]));  %正太分布，均值为0  方差为1
        end
    end
end
```

根据输入信息长度不同以及cover image的size不同，产生不同的水印。

实现基于Trellis Code的 E_BLK_8/D_BLK_8 系统

- 基于Trellis Code的E_BLK_8

```
function [wimg] = Trellis_E_BLK_8(img,message, wi, alpha)
[width, height] = size(img);
img = im2double(img);
w = (zeros(size(wi{1}))); % wi里是八个8*8矩阵
message = [message 0 0];  % 是否pad两个0
for i=1:size(message,2)
    if(message(i)==1)
       w = w + wi{i};
    elseif(message(i)==0)
        w = w - wi{i};
    end

end
w_mean = mean(mean(w));
w = w - w_mean;
w_std = std2(w);
w = w / w_std;

% extract mark from img before modi?cation
v = zeros(8,8);
nvo = zeros(8,8);
n = zeros(8,8);
for i=1:width
    for j=1:height
        imod8 = int32(mod(i-1,8))+1;
        jmod8 = int32(mod(j-1,8))+1;
        nvo(imod8, jmod8) = nvo(imod8, jmod8) + double(img(i,j));
        n(imod8, jmod8) = n(imod8, jmod8) + 1;
    end
```

```
end
v = nvo./n;

% use blind embedding to choose a new vector in mark space
% that is close to a given extracted mark and (hopefully) inside
% the detection region
vw = zeros(8,8);
vw = v + alpha*w;

% modify an image so that when entered into ExtractMark
% it will produce (approximately) a given mark
delta = n.*vw -nvo;
wimg = zeros(size(img));
for i=1:width
    for j=1:height
        imod8 = int32(mod(i-1,8))+1;
        jmod8 = int32(mod(j-1,8))+1;
        oldPixel = double(img(i,j));
        newPixel = double(img(i,j))+delta(imod8,jmod8)/n(imod8,jmod8);
        if(newPixel <0)
            img(i,j) = 0;
        elseif(newPixel>255)
            img(i,j) = 255;
        else
            img(i,j) = newPixel;
        end
        n(imod8,jmod8) = n(imod8,jmod8) - 1;
        delta(imod8,jmod8) = delta(imod8,jmod8) - (img(i,j)-oldPixel);
    end
end
wimg = img;
end
```

- 基于Trellis Code的D_BLK_8

```
function [wm, cc] = Trellis_D_BLK_8(wimg, wi, threshold)
[width,height] = size(wimg);
wimg = im2double(wimg);
% ExtractMark
nvo = zeros(8,8);
n = zeros(8,8);
for i=1:width
    for j=1:height
        imod8 = int32(mod(i-1,8))+1;
        jmod8 = int32(mod(j-1,8))+1;
        nvo(imod8, jmod8) = nvo(imod8, jmod8) + double(wimg(i,j));
        n(imod8, jmod8) = n(imod8, jmod8) + 1;
    end
end
v = nvo./n;

% TrellisDemodulate  解码
lc0 = zeros(1,8)-1; lc0(1) = 0;
m1 = zeros(8,8); m0 = zeros(8,8);
nextState = [1 2;3 4;5 6;7 8;1 2;3 4;5 6;7 8];
```

```
for i=1:10
    lc1 = zeros(1,8)-1;
    for state=1:8
        if(lc0(state) ~= -1)
            if i<=size(wi,2)
                lc = sum(sum(v.*wi{i}))/(8*8);
            else
                tmp = randn([8,8]);
                lc = sum(sum(v.*tmp))/(8*8);
            end
            next = nextState(state,1);
            if (lc1(next) == -1 || lc1(next) < lc0(state) - lc)
                lc1(next) = lc0(state) -lc;
                m1(next,:) = m0(state,:);
            end
            % We know that the last two bits of the message must be 0
            if i <= 8
                next = nextState( state, 2 );
                if (lc1(next) == -1 || lc1(next) < lc0(state) + lc)
                    lc1( next ) = lc0( state ) + lc;
                    m1( next,: ) = m0( state,: );
                    m1(next,i) = 1;
                end
            end
        end
    end
    lc0 = lc1;
    m0 = m1;
end
[~,bestState]  = max(lc0);
wm = m0(bestState,:);

% TrellisModulate
w = (zeros(size(wi{1}))); % wi里是八个8*8矩阵
% message = wm;
message = [wm 0 0];  % 是否pad两个0
for i=1:size(wi,2)
    if(message(i)==1)
       w = w + wi{i};
    elseif(message(i)==0)
        w = w - wi{i};
    end

end

% Correlation Coefficient
v_mean = mean(mean(v));
vw = sum(sum((v-v_mean).*(w-mean(mean(w)))));
vv = sum(sum((v-v_mean).*(v-v_mean)));
ww = sum(sum((w-mean(mean(w))).*(w-mean(mean(w)))));
if( abs(vv * ww) < 0.0001 )
    cc = 0;
else
    cc = vw / sqrt( vv * ww );
end

if((cc) < threshold)
    wm = [-1,-1,-1,-1,-1,-1,-1,-1];
```

```
79  end
80  end
```

注意message有填充两个0的区别。

实验分析与结论

问题2 使用一张水印测试基于 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

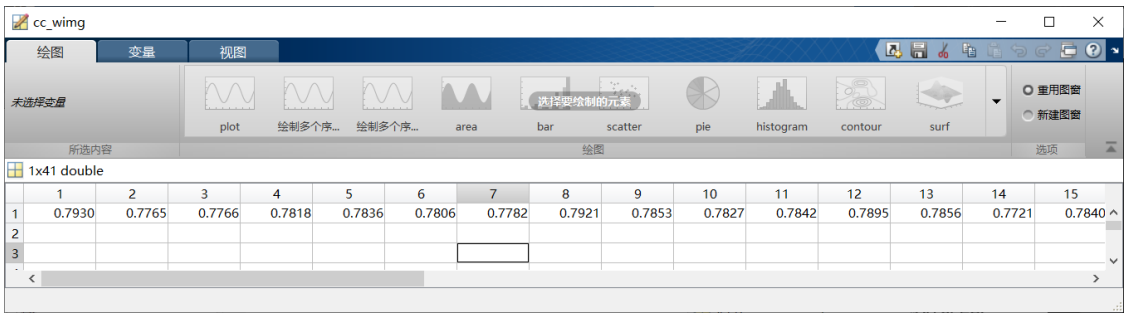
```
clc;
path = '../data/';
list = dir(path);
alpha = 2;
seed = 1;
message =  randi([0,1],[1,8]); disp(message);
array_wimg = [];cc_wimg = [];
array_img  = [];cc_img = [];
wi = GenerateWatermark(8,8,8); % wi is a cell.
for i=3:(size(list,1))
    img = imread([path,list(i).name]);
    wimg = Simple_E_BLK_8(img,message, wi,alpha);
    [array_wimg(i-2,:),cc_wimg(i-2)] = Simple_D_BLK_8(wimg,wi,0.5);
    [array_img(i-2,:),cc_img(i-2)] = Simple_D_BLK_8(img,wi,0.5); % 不添加水印直接decode
end
[FalsePositive,FalseNegative ] = getDetectRate(array_wimg, array_img);
```

代码运行后我们比较变量`array_wimg`， `array_img`：

- cc_img:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-0.0970	0.0450	0.0150	0.0137	-0.0098	0.0967	0.0945	-0.1646	0.0456	-0.0972	-0.1656	0.0122	0.0639	-0.0226	0.1156
2															
3															

- cc_wimg



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.7930	0.7765	0.7766	0.7818	0.7836	0.7806	0.7782	0.7921	0.7853	0.7827	0.7842	0.7895	0.7856	0.7721	0.7840
2															
3															

我们可以发现有水印和无水印的数值差距还是较为明显的，因此我们设置0.5作为阈值，当检测值`cc`的绝对值小于阈值时，判定该图片没有水印，否则就含有水印。

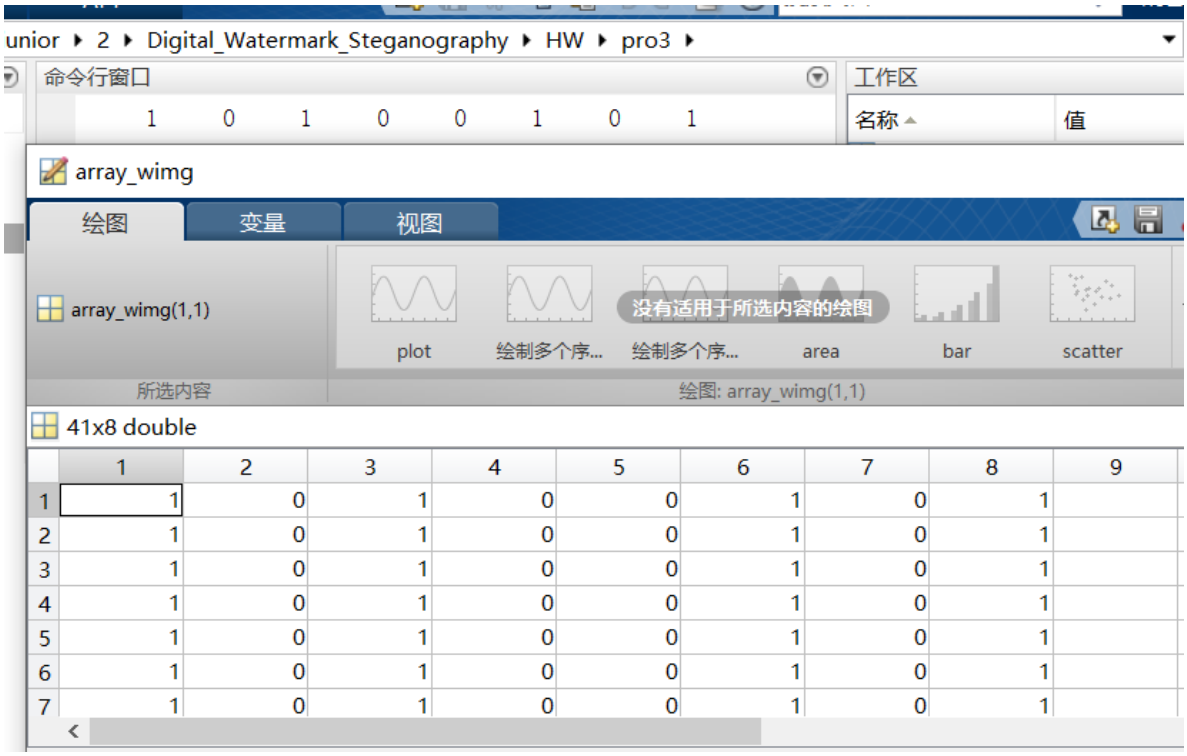
因此我用`getDetectRate`来获得False Positive/Negative Rate 的值。

```
function [FalsePositive,FalseNegative] = getDetectRate(array_wimg,
array_img)
FalsePositive = 0;
FalseNegative = 0;
for i=1:size(array_wimg,1)
    if array_wimg(i,:) == [-1,-1,-1,-1,-1,-1,-1,-1]
        FalseNegative = FalseNegative + 1;
    end
end

for i=1:size(array_img,1)
    if array_img(i,:) ~= [-1,-1,-1,-1,-1,-1,-1,-1]
        FalsePositive = FalsePositive + 1;
    end
end
% FalsePositive = FalsePositive/size(array_img,1);
% FalseNegative = FalseNegative/size(array_wimg,1);
end
```

最后我们得到在该水印下，嵌入随机的message时，我们False Positive Rate 为0/41和 False Negative Rate 为0/41. 可以看到该系统对于检测是否存在水印的准确率很高。

至于decode的准确率，在运行10次之后，发现解码的准确率是100%。



	1	2	3	4	5	6	7	8	9
1	1	0	1	0	0	1	0	1	
2	1	0	1	0	0	1	0	1	
3	1	0	1	0	0	1	0	1	
4	1	0	1	0	0	1	0	1	
5	1	0	1	0	0	1	0	1	
6	1	0	1	0	0	1	0	1	
7	1	0	1	0	0	1	0	1	

问题4 测试基于TrellisCode的E_BLK_8/D_BLK_8 系统应用于不同封面时的检测准确率

```
clc;
path = '../data/';
list = dir(path);
alpha = sqrt(8);
seed = 0;
message =  randi([0,1],[1,8]); disp(message);
array_wimg = [];cc_wimg = [];
array_img  = [];cc_img = [];
wi = GenerateWatermark(10,8,8); % wi is a cell.
```

```
for i=3:(size(list,1))
    img = imread([path,list(i).name]);
    wimg = Trellis_E_BLK_8(img,message, wi,alpha);
    [array_wimg(i-2,:),cc_wimg(i-2)] = Trellis_D_BLK_8(wimg,wi,0.5);
    [array_img(i-2,:),cc_img(i-2)] = Trellis_D_BLK_8(img,wi,0.5); % 不添加水印直接decode
end
[FalsePositive,FalseNegative ] = getDetectRate(array_wimg, array_img);
```

- 没有在message后面pad两个0时，水印为随机生成，阈值为0.5
  - message=00000111, false positive = 0/41， false negative = 0/41，decode信息的错误个数1/41， decode信息的错误bit数 1/(41*8).
  - message=11001010, false positive = 0/41， false negative = 0/41，decode信息的错误个数3/41， decode信息的错误bit数 3/(41*8).
  - message=10001000, false positive = 0/41， false negative = 0/41，decode信息的错误个数8/41， decode信息的错误bit数 8/(41*8).
  - message=10010000, false positive = 0/41， false negative = 0/41，decode信息的错误个数4/41， decode信息的错误bit数 4/(41*8).
  - message=10010010, false positive = 0/41， false negative = 0/41，decode信息的错误个数0/41， decode信息的错误bit数 0/(41*8).

  可以看到在没有pad两个0的时候对于检测图片是否有水印没有影响，但是对于信息解码会有影响，可以看到平均$16/255 = 6.27\%$ 的解码错误率，其中错误的信息都只错了1bit。
- 在message后面pad两个0时，水印为随机生成，阈值为0.5
  - message=01011101, false positive = 0/41， false negative = 0/41，decode accuracy = 41/41 .
  - message=10100011, false positive = 0/41， false negative = 0/41，decode accuracy = 41/41 .
  - message=10000101, false positive = 0/41， false negative = 0/41，decode accuracy = 41/41 .
  - message=11111011, false positive = 0/41， false negative = 0/41，decode accuracy = 41/41 .
  - message=10001001, false positive = 0/41， false negative = 0/41，decode accuracy = 41/41，

  可以看到pad两个0之后，整个水印的解码成功率大幅上升，在255次检测中没有一次出错。

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

在信息末尾可以提高检测的准确率。

因为最后的两位数是固定的00，所以我们能确定最后的两个bit走的路线，因为最后是按照路径计算内积和，因此正确路径上的最后两位0都正确的，所以会有较大的内积和。这带来的大的差异可以用来提高准确率。

问题5 比较基于不同系统，E_SIMPLE_8/D_SIMPLE_8 和基于 Trellis Code 的 E_BLK_8/D_BLK_8 系统的检测准确率

实验所用代码

```
clc;
path = '../data/';
list = dir(path);
alpha = 2;
array_wimg = [];cc_wimg = [];
```

```
array_img  = [];cc_img = [];
wi = GenerateWatermark(10,8,8); % wi is a cell.
for test = 1:5
    message =  randi([0,1],[1,8]);
    fprintf('-----------message:%s------------\n',mat2str(message));
    for i=3:(size(list,1))
        img = imread([path,list(i).name]);
        wimg = Simple_E_BLK_8(img,message, wi,alpha);
        [array_wimg(i-2,:),cc_wimg(i-2)] = Simple_D_BLK_8(wimg,wi,0.5);
        [array_img(i-2,:),cc_img(i-2)] = Simple_D_BLK_8(img,wi,0.5); % 不添加水印直接decode
    end
    [FalsePositive,FalseNegative ] = getDetectRate(array_wimg, array_img);
    [wrongnum,wrongbit] = getDecodeRate(array_wimg, message);
    fprintf('-------Based on E_SIMPLE_8/D_SIMPLE_8------------------\n');
    fprintf('FalsePositive number = %d/41 \t FalseNegative number = %d/41 \n',FalsePositive,FalseNegative );
    fprintf('wrong decoded message num = %d/41\t wrong decoded message bits = %d/(41*8)\n',wrongnum,wrongbit );

    for i=3:(size(list,1))
        img = imread([path,list(i).name]);
        wimg = Trellis_E_BLK_8(img,message, wi,alpha);
        [array_wimg(i-2,:),cc_wimg(i-2)] = Trellis_D_BLK_8(wimg,wi,0.5);
        [array_img(i-2,:),cc_img(i-2)] = Trellis_D_BLK_8(img,wi,0.5); % 不添加水印直接decode
    end
    [FalsePositive,FalseNegative ] = getDetectRate(array_wimg, array_img);
    [wrongnum,wrongbit] = getDecodeRate(array_wimg, message);
    fprintf('-------------Based on Trellis Code-----------------\n');
    fprintf('FalsePositive number = %d/41 \t FalseNegative number = %d/41 \n',FalsePositive,FalseNegative );
    fprintf('wrong decoded message num = %d/41\t wrong decoded message bits = %d/(41*8)\n\n\n',wrongnum,wrongbit );
end
```

我们固定每次的水印但是改变message的值。

实验结果

```
----------message:[0 0 1 1 0 1 1 1]------------
-------Based on E_SIMPLE_8/D_SIMPLE_8------------------
FalsePositive number = 1/41      FalseNegative number = 0/41
wrong decoded message num = 0/41     wrong decoded message bits = 0/(41*8)
-------------Based on Trellis Code------------------
FalsePositive number = 0/41      FalseNegative number = 41/41
wrong decoded message num = 41/41    wrong decoded message bits = 328/(41*8)


----------message:[1 0 0 1 1 0 0 0]------------
-------Based on E_SIMPLE_8/D_SIMPLE_8------------------
FalsePositive number = 1/41      FalseNegative number = 0/41
wrong decoded message num = 41/41    wrong decoded message bits = 123/(41*8)
-------------Based on Trellis Code------------------
FalsePositive number = 0/41      FalseNegative number = 38/41
wrong decoded message num = 38/41    wrong decoded message bits = 304/(41*8)


----------message:[0 0 0 1 0 1 1 1]------------
-------Based on E_SIMPLE_8/D_SIMPLE_8------------------
FalsePositive number = 1/41      FalseNegative number = 0/41
wrong decoded message num = 0/41     wrong decoded message bits = 0/(41*8)
-------------Based on Trellis Code------------------
FalsePositive number = 0/41      FalseNegative number = 0/41
wrong decoded message num = 0/41     wrong decoded message bits = 0/(41*8)


----------message:[1 1 0 1 0 1 1 0]------------
-------Based on E_SIMPLE_8/D_SIMPLE_8------------------
FalsePositive number = 1/41      FalseNegative number = 4/41
wrong decoded message num = 41/41    wrong decoded message bits = 106/(41*8)
-------------Based on Trellis Code------------------
FalsePositive number = 0/41      FalseNegative number = 0/41
wrong decoded message num = 0/41     wrong decoded message bits = 0/(41*8)


----------message:[0 0 1 1 0 1 1 0]------------
-------Based on E_SIMPLE_8/D_SIMPLE_8------------------
FalsePositive number = 1/41      FalseNegative number = 0/41
wrong decoded message num = 0/41     wrong decoded message bits = 0/(41*8)
-------------Based on Trellis Code------------------
FalsePositive number = 0/41      FalseNegative number = 0/41
wrong decoded message num = 0/41     wrong decoded message bits = 0/(41*8)
```

我们可以看到两个系统的检测准确率其实都很高，false neg/pos的值很小，但是可以发现基于trellis code的系统在解码的正确率上要胜过基于SIMPLE_8的。

原因：

第一个系统只有简单的encode和decode，第二个系统会有错误码的恢复，找到当前最有可能的一个码字，因此显然是后一个的准确率会更高一点。

实验感想

其实搞清楚trellis code花了我很长时间，很不容易写完以后，又发现程序有bug。debug到天昏地暗，最后实在没办法，就把书上附录里面的c代码直接copy过来，然后控制变量，一步一步比对。最后才发现一个类型转换和一个迭代关系上的bug。只能说debug太不容易了，写完这份报告我终于能去复习了。

体会了E_BLK_8/D_BLK_8 系统对于检测图片是否带水印有很大优势，也算是比较明白了trellis code 的算法。