武汉大学国家网络安全学院

信息隐藏实验报告

学　　号　2021302181156

姓　　名　赵伯侯

实验名称　安全隐写对抗技术

指导教师　任延珍

一、 实验名称: 安 全 隐 写 对 抗 技 术

二、 实验目的:

1. 实现 LSBM 隐写算法的消息嵌入和提取并分析其安全性

2. 实现卡方分析，并且利用卡方分析检测不同嵌入率下 LSBR 和 LSBM 的情况，进行对比分析

3. 实现 F4 隐写算法的消息嵌入和提取并分析其安全性

4. 实现 F5 隐写算法的消息嵌入和提取并分析其安全性

三、 实验原理:

（一）LSBM 隐写算法

为了克服在 LSBR 算法中产生的值对效应，当载体图像的 LSB 与消息比特不同时，将像素值随机加减 1，且为了防止溢出，当载体像素值为 0 时，则进行加 1 操作，若像素值为 255，则进行减 1 操作，对于其他的载体像素值则随机进行加 1 或者减 1。从而避免偶数只能加 1 变成奇数和奇数只能减 1 变成偶数的值对效应。

（二）卡方分析

针对图像像素 LSB 全嵌入的情况，利用数理统计假设检验中的卡方检验模型来分析统计样本的实际观测值与理论推断值之间的偏离程度，决定卡方值的大小，如果卡方值越大，二者的偏差程度越大，反之，二者偏差越小；若两者完全相等时，卡方值就为 0，表明理论值完全符合

设图像中灰度值为 i 的像素数量为 h_i, 其中 i 的取值为 $[0, 127]$

对于任意一组奇偶灰度值 2i 和 2i+1，其对应的像素块数目为 h_{2i} 和 h_{2i+1}, 将任意一组奇偶灰度值的平均值设置为 y_1, 将二者的差定义为 y_2 通过比较 y_1 和 y_2 之间的偏离程度来计算卡方统计量，

由此可得定义变量 $t=\sum_{i=1}^{127} \frac{(h2i-1-h2i)^2}{2\times(h2i-1+h2i)}$

t 满足自由度为 v=d-1 的卡方分布, 由此可以计算得到自由度为 V 的卡方分布在 x 点处的分布函数值即 $F(x < X)$

为了使结果更加明显定义变量 p 代表当前图像被隐写概率 $p = 1 - F(x < X)$

同时为了能够分析出图像的嵌入率，在计算卡方分析时需要将图像按行分成 10 份，每 10% 计算一次 p，通过观察 p 在何时值变小来确定图像的嵌入率

（三）F4 隐写算法

用秘密信息比特替换 JPEG 图像量化后的 DCT 系数的 LSB，但不处理为 0 的 DCT 系数，从而保证在隐写后人眼无法分辨隐写图和原图的差别。

为了避免在 jsteg 隐写过程中出现的值对效应和在 F3 隐写中出现的偶系数增多的缺点，在 F4 中使用全新的替换策略：在提取时只需要将 AC 系数中的非 0 位正奇数和负偶数代表 1，正偶数和负奇数代表 0。

（四）F5 隐写算法

F5 隐写算法用秘密信息比特替换 JPEG 图像量化后的 DCT 系数的 LSB，但不处理为 0 的 DCT 系数，从而保证在隐写后人眼无法分辨隐写图和原图的差别。

另外 F5 隐写算法使用了矩阵编码即汉明码的方法，在嵌入过程中使用到了汉明码的一致校验阵，嵌入规则采用与 F4 相同的嵌入规则。矩阵编码的目的就是通过编码的方式减少每个载体分组中所需要的嵌入修改次数，提高嵌入效率，使每个 LSB 修改可以嵌入更多的秘密比特。

在嵌入时首先选取 7 位载体矩阵 a，将其与汉明码的一致校验阵进行矩阵乘法运算得到 c，将 c 按位模 2 后将其视为 8 进制转换为 10 进制，得到将要修改的位 r，若 r=0 则不需要对载体矩阵 a 进行修改，若 r 不为 0 则修改载体矩阵 a 中的第 r 位得到修改后的载体矩阵。

在提取时将修改后的载体矩阵与汉明码一致校验阵进行矩阵乘法运算后即可得到隐藏的信息。

在进行位置置乱时采用 Arnold 变换对隐写前的图像进行在有限区域内进行反复折叠、拉伸变换，该变换采用的算法如下所示

$$\begin{bmatrix} yn+1 \\ xn+1 \end{bmatrix} = \begin{bmatrix} 1 & b \\ a & ab+1 \end{bmatrix} \begin{bmatrix} yn \\ xn \end{bmatrix}$$

在隐写结束后以及 DCT 逆变换之前，需要将图像的位置置乱进行还原，位置置乱的还原算法如下所示

$$\begin{bmatrix} yn+1 \\ xn+1 \end{bmatrix} = \begin{bmatrix} ab+1 & -b \\ -a & 1 \end{bmatrix} \begin{bmatrix} yn \\ xn \end{bmatrix}$$

四、 实验内容:

（一）实验 1.1：LSBM 隐写算法

LSBM 信息隐藏的代码如代码 1 所示

```
input="Lena.bmp";    %隐写载体
output="scover_LSBM.bmp";     %隐写后图像保存位置
k=0.9;  %嵌入率

cover = imread(input) ;
ste_cover = cover;
ste_cover = double( ste_cover) ;

[ m, n] = size( ste_cover) ;

%按照嵌入率生成对应数量的秘密信息
x=round(m*k);
msg=randsrc(x,n,[0 1;0.5 0.5]);
writematrix(msg,"message.txt");

change=0;      %记录隐写时修改的次数

% 在 LSB位进行隐写
for i = 1:x
    for j=1:n
        if mod(ste_cover(i,j),2)~=msg(i,j)
```

```
            change=change+1;
            if ste_cover(i,j)==0
                ste_cover(i,j)=1;
            elseif ste_cover(i,j)==255
                ste_cover(i,j)=254;
            else
                ste_cover(i,j)=plus_or_sub(ste_cover(i,j));
            end
        end
    end
end

% 隐写后的图像输出
ste_cover = uint8( ste_cover) ;
imwrite( ste_cover, output) ;

% 显示实验结果
subplot( 2, 2, 1) ; imshow( cover) ; title( "原始图像") ;
subplot( 2, 2, 2) ; imshow( ste_cover) ; title("隐藏信息的图像") ;
subplot( 2, 2, 3) ; imhist( cover) ; title( "原始图像直方图") ;
subplot( 2, 2, 4) ; imhist( ste_cover) ; title("隐藏信息的图像直方图") ;
```

代码 1: LSBM 隐写代码

LSBM 信息提取的代码如代码 2 所示

```
input="scover_LSBM.bmp";     %隐写后图像
goalfile="result.txt";     %提取信息存放位置
k=0.9;                     %嵌入率

ste_cover = imread( input) ;
[m,n] = size( ste_cover) ;
```

```
x=round(m*k);

len_total=x*n;
result=zeros(x,n);

%p代表提取信息的个数
p = 1;
for i = 1:x
    for j = 1:n
        if mod( ste_cover( i, j) , 2) == 1
            result(i,j)=1;
        else
            result(i,j)=0;
        end
        if p == len_total
            break;
        end
        p = p + 1;
    end
    if p == len_total
        break;
    end
end

%将提取得到的信息写入文件
writematrix(result,goalfile);
```

代码 2: LSBM 提取代码

LSBM 隐写过程中使用的随机加减的函数代码如代码 3 所示

```
%随机生成一个+1或-1
function result=plus_or_sub(input)
x=randsrc;
result=input+x;
```

代码 3: LSBM 隐写随机加减代码

（二）实验 **1.2**：卡方分析

卡方分析代码如代码 4 所示

```
image="scover_LSBM.bmp";
%image="scover_LSBR.bmp";
%image="F4_scover.png";
%image="F5_scover.png";

%读取图像信息若为png图片则只取其红色部分
ste_cover=imread(image);
ste_cover=ste_cover(:,:,1);

[m,n]=size(ste_cover);

%存放每一份图像的卡方统计量和被隐写概率
S=[];
P_values=[];

%将图像按10%的分割，一共分成10份
part=m/10;

%统计每一个10%的卡方统计量
for j=0:9
```

```
    h=imhist(ste_cover(floor(j* part)+1:floor((j+1)* part),:));
    h(1) = 0;
    h(2) = 0;
    h_length=size(h);
    p_num=floor(h_length/2);
    Spov=0; %记录卡方统计量
    K=0;
    for i=1:p_num
        if(h(2*i-1)+h(2*i))~=0
            Spov=Spov+(h(2*i-1)-h(2*i))^2/(2*(h(2*i-1)+h(2*i)));
            K=K+1;
        end
    end
    %Spov为卡方统计量，K-1为自由度
    P=1-chi2cdf(Spov,K-1);

    if j~=0
      Spov=Spov+S(j); %若不注释则为累计卡方统计量
    end

    S=[S Spov];
    P_values=[ P_values P];
end

%显示变化曲线，x_label是横坐标，代表分析样本占整幅图像的百分比
x_label=0.1:0.1:1;
figure,
plot(x_label,P_values,'LineWidth',2),title('p值与分析图像的像素比例关系');
xlabel('分析图像的像素比例');
```

```
ylabel('p值');
ylim([0, 1])
```

代码 4: k2 分析代码

在卡方分析中用到的 LSBR 隐写的算法如代码 5 所示

```
input="Lena.bmp";    %隐写载体
output="scover_LSBR.bmp";     %生成载密图像
k=0.9;  %嵌入率

cover = imread(input) ;
ste_cover = cover;
ste_cover = double( ste_cover) ;

[ m, n] = size( ste_cover) ;

%按照嵌入率的大小随机生成生成对应的秘密信息01矩阵
x=round(m*k);
msg=randsrc(x,n,[0 1;0.5 0.5]);
writematrix(msg,"message.txt");

change=0;      %统计嵌入信息时的修改次数

% 在 LSB位隐写
for i = 1:x
    for j=1:n
        change=change+1;
        ste_cover(i,j) = ste_cover(i,j) -mod( ste_cover(i,j) , 2) + msg(i,j);
    end
end

```

```
% 隐写后的图像输出
ste_cover = uint8( ste_cover) ;
imwrite( ste_cover, output) ;

% 显示实验结果
subplot( 2, 2, 1) ; imshow( cover) ; title( "原始图像") ;
subplot( 2, 2, 2) ; imshow( ste_cover) ; title("隐藏信息的图像") ;
subplot( 2, 2, 3) ; imhist( cover) ; title( "原始图像直方图") ;
subplot( 2, 2, 4) ; imhist( ste_cover) ; title("隐藏信息的图像直方图") ;
```

代码 5: LSBR 隐写代码

在卡方分析中用到的 LSBR 提取的算法如代码 6 所示

```
input="scover_LSBR.bmp";     %生成的载密图像
goalfile="result.txt";     %提取得到的秘密信息保存位置
k=0.9;                     %嵌入率

ste_cover = imread( input) ;
[m,n] = size( ste_cover) ;
x=round(m*k);

len_total=x*n;
result=zeros(x,n);

%p代表提取到秘密信息的bit位
p = 1;
for i = 1:x
    for j = 1:n
        if mod( ste_cover( i, j) , 2) == 1
            result(i,j)=1;
        else
```

```
            result(i,j)=0;
        end
        if p == len_total
            break;
        end
        p = p + 1;
    end
    if p == len_total
        break;
    end
end

%将提取得到的秘密信息写入文件
writematrix(result,goalfile);
```

代码 6: LSBR 提取代码

（三）实验 2.1：F4 隐写算法

F4 隐写算法的代码如代码 7 所示

```
input_path='lena.png';
output_path='F4_scover.png';

%读取隐写载体的红色部分
A=imread(input_path);
R=A(:,:,1);
G=A(:,:,2);
B=A(:,:,3);

[m,n]=size(R);

```

```
%获取隐藏信息的二进制数据
f_id = fopen("message.txt", "r") ;
[ msg,len_total] = fread( f_id, 'ubit1') ;

%只取R层进行隐写
I =R;

%量化矩阵
mask1=[16 11 10 16 24 40 51 61
       12 12 14 19 26 58 60 55
       14 13 16 24 40 57 69 56
       14 17 22 29 51 87 80 62
       18 22 37 56 68 109 103 77
       24 35 55 64 81 104 113 92
       49 64 78 87 103 121 120 101
       72 92 95 98 112 100 103 99];

% 将图像进行jepg压缩
jpeg_img=zeros(m,n);
for row=1 :8:m
    for col= 1:8:n
        DCT_matrix=I(row:row+7,col:col+7);
        %将每一块进行DCT变换
        DCT_matrix=dct2(DCT_matrix);
        %将每一块通过矩阵进行量化
        DCT_matrix=round(DCT_matrix./mask1);
        %将量化后的块写入jpeg_img
        jpeg_img(row:row+7,col:col+7)=DCT_matrix;
    end
```

```
end

%显示隐写前的DCT系数直方图
subplot(2,2,3),histogram(jpeg_img),title('经过隐写前的DCT系数图');
%subplot(2,1,1),histogram(jpeg_img),title('经过隐写前的DCT系数图');

%隐写
change=0;    %记录修改载体次数5
x=1;  %当前处理的字节指针
for i=1:m
    for j=1:n
        if x<=len_total && (mod(i-1,8)~=0 || mod(j-1,8)~=0) && (i~=1 || j~=1)
              %只取LSB位
            ac=jpeg_img(i,j);
            if ac~=0 %去掉0
                if mod(ac,2)==0
                    if ac>=0
                        if msg(x)==1
                            ac=ac-1;
                        end
                    else
                        if msg(x)==0
                            ac=ac+1;
                        end
                    end
                else
                    if ac>=0
                        if msg(x)==0
                            ac=ac-1;
```

```
                        end
                    else
                        if msg(x)==1
                            ac=ac+1;
                        end
                    end
                end
                jpeg_img(i,j)=ac;
                change=change+1;
                %若修改后ac系数为0则重新对该位进行隐写
                if ac~=0
                    x=x+1;
                end
            end
        end
    end
end

%显示隐写之后的DCT系数直方图
subplot(2,2,4),histogram(jpeg_img),title('经过隐写后的DCT系数图');
%subplot(2,1,2),histogram(jpeg_img),title('经过隐写后的DCT系数图');

% 逆DCT变换
for row=1:8:m
    for col=1:8:n
        IDCT_matrix=jpeg_img(row:row+7,col:col+7);
        IDCT_matrix=round(idct2(IDCT_matrix.*mask1));
        recon_img(row:row+7,col:col+7)=IDCT_matrix;
    end
```

```
end

%将三个颜色部分混合
Mix=cat(3,recon_img,G,B);
imwrite(Mix,output_path);

figure(1);
subplot(2,2,1),imshow(A),title('原图');
subplot(2,2,2),imshow(Mix),title('嵌入信息');
```

代码 7: F4 隐写代码

F4 隐写信息提取的代码如代码 8 所示

```
input_path='F4_scover.png';    %载密图像位置
result_path='result.txt';
len_total=840;

%读取图像后只取R层进行提取
A=imread(input_path);
R=A(:,:,1);
G=A(:,:,2);
B=A(:,:,3);
I=R;

[m,n]=size(R);

frr = fopen(result_path, 'a');

%量化矩阵
mask1=[16 11 10 16 24 40 51 61
    12 12 14 19 26 58 60 55
```

```
    14 13 16 24 40 57 69 56
    14 17 22 29 51 87 80 62
    18 22 37 56 68 109 103 77
    24 35 55 64 81 104 113 92
    49 64 78 87 103 121 120 101
    72 92 95 98 112 100 103 99];

% 将图像进行jepg压缩
jpeg_img=zeros(m,n);
for row=1 :8:m
    for col= 1:8:n
        DCT_matrix=I(row:row+7,col:col+7);
        %将每一块进行DCT变换
        DCT_matrix=dct2(DCT_matrix);
        %将每一块通过矩阵进行量化
        DCT_matrix=round(DCT_matrix./mask1);
        %将量化后的块写入jpeg_img
        jpeg_img(row:row+7,col:col+7)=DCT_matrix;
    end
end

%提取
x=1;  %当前处理的字节指针
msg=zeros(len_total,1);
for i=1:m
    for j=1:n
        if x<=len_total && (mod(i-1,8)~=0 || mod(j-1,8)~=0) &&(i~=1 || j~=1) %
            只取LSB位
            ac=jpeg_img(i,j);
```

```
47              if ac~=0   %去掉0
48                  if ac>0
49                      if mod(ac,2)==0
50                          fwrite( frr, 0,'ubitl') ;
51                          msg(x)=0;
52                      else
53                          fwrite( frr, 1,'ubitl') ;
54                          msg(x)=1;
55                      end
56                  else
57                      if mod(ac,2)==0
58                          fwrite( frr, 1,'ubitl') ;
59                          msg(x)=1;
60                      else
61                          fwrite( frr, 0,'ubitl') ;
62                          msg(x)=0;
63                      end
64                  end
65                  x=x+1;
66
67              end
68          end
69      end
70  end
71  fclose(frr) ;
```

代码 8: F4 提取代码

（四）实验 **2.2**：**F5** 隐写算法

F5 隐写算法的代码如代码 9 所示

```
image_path="lena.png";     %载体图像
scover_path="F5_scover.png";     %隐写后输出载密图像
message_path="message.txt";

%位置置乱函数参数
times=10;      %置乱次数
c_a=3;
c_b=5;

%读取秘密信息
f_id = fopen(message_path, "r") ;
[ msg,len_total] = fread( f_id, 'ubit1') ;
data_len=len_total/3;

%读取载体图像，只取其红色部分进行隐写
image=imread(image_path);
R=image(:,:,1);
G=image(:,:,2);
B=image(:,:,3);
cover=R;

% 标准量化表
Q=[16 11 10 16 24 40 51 61
   12 12 14 19 26 58 60 55
   14 13 16 24 40 57 69 56
   14 17 22 29 51 87 80 62
   18 22 37 56 68 109 103 77
```

```
   24 35 55 64 81 104 113 92
   49 64 78 87 103 121 120 101
   72 92 95 98 112 100 103 99];

%汉明码一致校验矩阵
M=[0 0 0 1 1 1 1
   0 1 1 0 0 1 1
   1 0 1 0 1 0 1];

% 初始化，DCT转换和量化
[h,w]=size(cover);
D=zeros(h,w);            %零时存储矩阵
for i=1:h/8
    for j=1:w/8
        D(8*(i-1)+1:8*i,8*(j-1)+1:8*j)=dct2(cover(8*(i-1)+1:8*i,8*(j-1)+1:8*j))
            ;
        D(8*(i-1)+1:8*i,8*(j-1)+1:8*j)=round(D(8*(i-1)+1:8*i,8*(j-1)+1:8*j)./Q)
            ;
    end
end

%位置置乱操作
%figure(1);imshow(D)
N=h;
D2=zeros(h,w);
for i=1:times
    for x=1:h
        for y=1:w
            xx=mod((x-1)+c_b*(y-1),N)+1;    %mod(a,b)就是a除以b的余数
```

```
            yy=mod(c_a*(x-1)+(c_a*c_b+1)*(y-1),N)+1;
            D2(yy,xx)=D(y,x);
        end
    end
    D=D2;
end
%figure(2);imshow(D);

% 隐写
stego=D;
num=1;          %记录目前已经嵌入的data数量
a=zeros(7,1);   %记录隐写载体的7个可隐写位
k=1;            %记录当前7个数值中已取数量
sit=zeros(7,2); %记录当前7个数在载体中的位置
change=0;       %记录发生修改的次数

for i=1:h
    for j=1:w
        if (mod(i-1,8)~=0 || mod(j-1,8)~=0) && (i~=1 || j~=1)
            if (D(i,j)>0 && mod(D(i,j),2)==1)||(D(i,j)<0 && mod(D(i,j),2)==0)
                    %正奇数或负偶数为1
                a(k)=1;
                sit(k,1)=i;
                sit(k,2)=j;
                k=k+1;
            elseif (D(i,j)<0 && mod(D(i,j),2)==1)||(D(i,j)>0 && mod(D(i,j),2)
                ==0)    %负奇数或正偶数为0
                a(k)=0;
                sit(k,1)=i;
```

```
                sit(k,2)=j;
                k=k+1;
            end
        end

        %当有7个隐写载体可隐写位时
        if(k>7)
            %据算出n的值，n为要修改的位数
            data_bit=[msg(num*3-2);msg(num*3-1);msg(num*3)];
            temp=M*a;
            temp=mod(temp,2);
            n=bitxor(data_bit,temp);      %按位异或
            n=n(1)*4+n(2)*2+n(3);

            % 修改第n位的DCT值
            if n>0
                if D(sit(n,1),sit(n,2))<0
                    stego(sit(n,1),sit(n,2))=D(sit(n,1),sit(n,2))+1;
                    change=change+1;
                elseif D(sit(n,1),sit(n,2))>0
                    stego(sit(n,1),sit(n,2))=D(sit(n,1),sit(n,2))-1;
                    change=change+1;
                end

                % 检查修改过后的DCT值是否为0，若为0则重新选择1位数据作为载体信
                    号
                if stego(sit(n,1),sit(n,2))==0
                    k=k-1;
                    sit(n:k-1,:)=sit(n+1:k,:);
```

```
                    a(n:k-1)=a(n+1:k);
                    continue;
                end
            end
            num=num+1;
            k=1;
        end
        if(num>data_len)
            break;
        end
    end
    if(num>data_len)
        break;
    end
end

%位置置乱还原
%figure(3);imshow(stego);
stego2=stego;
for i=1:times
    for x=1:h
        for y=1:w
            xx=mod((c_a*c_b+1)*(x-1)-c_b*(y-1),N)+1;
            yy=mod(-c_a*(x-1)+(y-1),N)+1  ;
            stego2(yy,xx)=stego(y,x);
        end
    end
    stego=stego2;
end
```

```
%figure(4);imshow(stego);

% 逆DCT转换，转换成伪装图像
for i=1:h/8
    for j=1:w/8
        stego(8*(i-1)+1:8*i,8*(j-1)+1:8*j)=stego(8*(i-1)+1:8*i,8*(j-1)+1:8*j).*
            Q;
        stego(8*(i-1)+1:8*i,8*(j-1)+1:8*j)=idct2(stego(8*(i-1)+1:8*i,8*(j-1)
            +1:8*j));
    end
end

%隐写后图像与其余两个颜色部分混合后保存
Mix=cat(3,stego,G,B);
imwrite(Mix,scover_path);

figure(5);
subplot(1,2,1),imshow(image),title('原图');
subplot(1,2,2),imshow(Mix),title('嵌入信息后的图像');
```

代码 9: F5 隐写代码

F5 隐写信息提取的代码如代码 10 所示

```
image_name="F5_scover.png";      %隐写后图像
len_total=840;                   %提取的比特长度
result_path="result.txt";

%位置置乱参数
times=10;     %置乱次数
c_a=3;
c_b=5;
```

```
9
10 data_len=len_total/3;  %每三个比特分一组，总的组数
11
12 %读取载体图像，只取其红色部分进行提取
13 image=imread(image_name);
14 R=image(:,:,1);
15 G=image(:,:,2);
16 B=image(:,:,3);
17 stego=R;
18
19 % 初始化标准量化表
20 Q=[16 11 10 16 24 40 51 61
21    12 12 14 19 26 58 60 55
22    14 13 16 24 40 57 69 56
23    14 17 22 29 51 87 80 62
24    18 22 37 56 68 109 103 77
25    24 35 55 64 81 104 113 92
26    49 64 78 87 103 121 120 101
27    72 92 95 98 112 100 103 99];
28
29 [h,w]=size(stego);
30
31 %汉明码一致校验矩阵
32 M=[0 0 0 1 1 1 1
33    0 1 1 0 0 1 1
34    1 0 1 0 1 0 1];
35
36 D=zeros(h,w);
37
```

```
% DCT转换和量化
for i=1:h/8
    for j=1:w/8
        D(8*(i-1)+1:8*i,8*(j-1)+1:8*j)=dct2(stego(8*(i-1)+1:8*i,8*(j-1)+1:8*j))
            ;
        D(8*(i-1)+1:8*i,8*(j-1)+1:8*j)=round(D(8*(i-1)+1:8*i,8*(j-1)+1:8*j)./Q)
            ;
    end
end

% 位置置乱
N=h;
D2=zeros(h,w);
for i=1:times
    for x=1:h
        for y=1:w
            xx=mod((x-1)+c_b*(y-1),N)+1;   %mod(a,b)就是a除以b的余数
            yy=mod(c_a*(x-1)+(c_a*c_b+1)*(y-1),N)+1;
            D2(yy,xx)=D(y,x);
        end
    end
    D=D2;
end

% 数据提取
num=1;          %记录目前已经嵌入的data数量
a=zeros(7,1);
k=1;            %记录当前7个数值中已取数量
result=zeros(len_total,1);
```

```
65
66  for i=1:h
67      for j=1:w
68          %取得隐写后载体a的7位值
69          if (mod(i-1,8)~=0 || mod(j-1,8)~=0) && (i~=1 || j~=1)
70              if (D(i,j)>0 && mod(D(i,j),2)==1)||(D(i,j)<0 && mod(D(i,j),2)==0)
                            %正奇数或负偶数为1
71                  a(k)=1;
72                  k=k+1;
73              elseif (D(i,j)<0 && mod(D(i,j),2)==1)||(D(i,j)>0 && mod(D(i,j),2)
                    ==0)    %负奇数或正偶数为0
74                  a(k)=0;
75                  k=k+1;
76              end
77          end
78
79          %表示集满7个数据
80          if k>7
81              temp=M*a;
82              temp=mod(temp,2);
83              result(num*3-2)=temp(1);
84              result(num*3-1)=temp(2);
85              result(num*3)=temp(3);
86              num=num+1;
87              k=1;
88          end
89          if num>data_len
90              break;
91          end
```

```
92      end
93      if num>data_len
94          break;
95      end
96  end
97
98  %提取到的秘密信息写入文件
99  frr = fopen(result_path, 'a');
100 for x=1:len_total
101     if result(x)==0
102         fwrite( frr, 0,'ubitl') ;
103     else
104         fwrite( frr, 1,'ubitl') ;
105     end
106 end
```

代码 10: F5 隐写信息提取代码

五、 实验环境:

1.win11 操作系统

2.MATLAB R2022a

六、 实验步骤:

（一）实验 **1.1**：**LSBM** 隐写算法

6.1.1 生成随机秘密消息

通过给出的嵌入率生成占到整个图像 k 倍的随机秘密信息矩阵，该矩阵通过调用函数 ransrc 生成。

6.1.2 在 LSB 位嵌入图像

遍历载体矩阵 LSB 位上的信息，如果与隐藏信息对应位的信息相同则不进行操作，如果与隐藏信息不同则判断载体上的像素值是否为 0 或 225，如果是 0 则将其 +1，如果是 255 则将其减 1。如果像素值不为 0 或 255 则调用随机加减函数将该像素值进行随机加减 1 的操作。

6.1.3 绘制灰度图像直方图

调用函数 imhist 处理原灰度图和经过隐写后的灰度图像，对比隐写前后图像的值对效应。

6.1.4 隐藏信息的提取

在隐藏信息提取时，只需要注意在隐写后的图像的像素值若为奇数则提取 1，若为偶数则提取 2。

（二）实验 **1.2**：卡方分析

6.2.1 载体图像分块

在进行卡方分析前，首先要对载体图像进行分块，在本次实验中将图像分成了 10 个小块，在之后的实验步骤中对每一块循环遍历计算相应的被隐写的概率

6.2.2 计算卡方统计量

在每一个小块中统计载体的灰度值为 i 的像素块的数量记录为 h(i) , 其中 i 的取值范围为 $0 \leq i \leq 127$。

由此可得统计量 $x_i = \frac{(h2i-1-h2i)^2}{2\times(h2i-1+h2i)}$ 服从自由度为 1 的卡方分布。

将所有的 x_i 累加得到卡方统计值

6.2.3 计算被隐写概率

调用函数 chi2cdf（X，V）计算自由度为 V 的卡方分布在 X 点处的分布函数值即 $F(x < X)$

最后统计 p=1-$F(x < X)$ 的值作为当前小块的被隐写概率

6.2.4 绘制每一个小块的被隐写概率图像

最后将每一次循环遍历得到的小块的被隐写概率绘制成图像的形式观察该折线图的拐点即为图像的嵌入率

（三）实验 2.1：F4 隐写算法

6.3.1 载体图像分块做二维 DCT 变换

首先与 DCT 隐写相同，先规定量化矩阵的值，然后将图像划分成 8*8 的块，对每一块先调用 dct2 函数进行 DCT 变换，然后将其与量化矩阵做矩阵除法，将得到的结果取整数，得到的即为经过处理的 DCT 系数，

6.3.2 获取分块中的 AC 系数并进行信息隐写

将上一个步骤中得到的经过处理的 DCT 系数取出 LSB 位，即去掉每一块中最左上角的块，剩余的系数即为 AC 系数，在 AC 系数中进行信息隐写为了解决隐写过程中产生的值对效应和载密图像中偶系数增多的问题，在修改 AC 系数时采用如下规则进行信息的嵌入 1. 跳过 0 2. 若 AC 系数为 2i，且 $i \geq 0$, 秘密信息比特为 0，则该 AC 系数不改变

3. 若 AC 系数为 2i，且 $i \geq 0$, 秘密信息比特为 1，则该 AC 系数变为 2i-1

4. 若 AC 系数为 2i，且 $i < 0$, 秘密信息比特为 0，则该 AC 系数变为 2i+1

5. 若 AC 系数为 2i，且 $i < 0$, 秘密信息比特为 1，则该 AC 系数不改变

6. 若 AC 系数为 2i+1，且 $i \geq 0$, 秘密信息比特为 0，则该 AC 系数变为 2i

7. 若 AC 系数为 2i+1，且 $i \geq 0$, 秘密信息比特为 1，则该 AC 系数不改变

8. 若 AC 系数为 2i+1，且 $i < 0$, 秘密信息比特为 0，则该 AC 系数不改变

9. 若 AC 系数为 2i+1，且 $i < 0$, 秘密信息比特为 1，则该 AC 系数变为 2i+2

10. 若 AC 系数变为 0，则该秘密信息比特重新隐藏

将该规则显式表示出来如图 1 所示

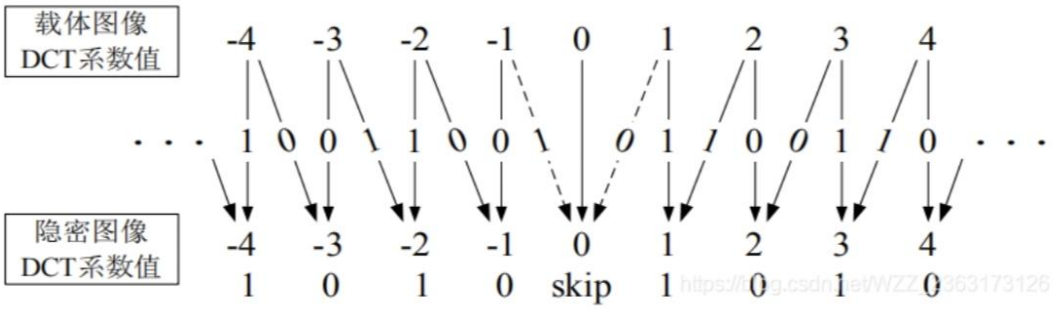

图 1: F4 隐写规则

6.3.3 隐写后的分块逆 DCT 变换还原

在隐写结束后对每一个分块调用 idct2 函数进行 DCT 还原，再将原图和隐写后的图片进行输出，比较两者之间的差别。

6.3.4 F4 隐写信息的提取

对于 F4 隐写信息后的图像，首先对其进行 DCT 变换，再比较每一个非零块，若为正奇数和负偶数则提取出 1，若为正偶数和负奇数则提取出 0，完成对隐写信息的提取。

（四）实验 **2.2**：**F5** 隐写算法

6.4.1 获取嵌入域

对输入图片的格式进行修改，在本次实验中使用的是 png 格式的图片，该种类型的图片大小为 m*n*3，只取出其红色部分的像素块当作信息隐写的载体，在信息隐写结束再将隐写后的红色部分与其他两个色域进行混合最后输出结果。

6.4.2 位置置乱

首先指定位置置乱的密钥即为 c_a,c_b,times 的值，其中 a 和 b 为在置乱算法中要用到的参数，times 为进行置乱的次数置乱时需要经过 times 次置乱循环，在每一次的循环中，遍历置乱前矩阵的各点的坐标 x,y 的值，通过运算算法

$$xx = mod((x-1) + c_b * (y-1), N) + 1$$

$$yx = mod(c_a * (x-1) + (c_a * c_b + 1) * (y-1), N) + 1$$

计算出该点的新坐标后将该点转移到新坐标的位置，在遍历完成后再继续进行下一轮的循环

6.4.3 编码参数确定

在本次实验中，选取每次嵌入信息的长度为 3，因为选取的汉明码一致校验阵的大小为 3*7, 所以选择每次嵌入信息的载体长度为 7，由此可以得到较高的嵌入效率

6.4.4 嵌入信息

开始嵌入信息后首先寻找 JPEG 图像中 AC 系数的非零值，在找到一个后将其坐标保存到变量 sit 中，sit 变量保存着当前 7 个隐写载体的位置坐标。

然后判断 AC 系数的值，若 AC 系数为正奇数或负偶数则在 a 矩阵的对应位置保存一个 1；若 AC 系数为负奇数或正偶数则在 a 矩阵的对应位置保存一个 0，a 矩阵中保存着 7 个隐写载体所代表的值。

在有 7 个隐写载体可隐写位时开始对载体进行隐写操作，首先计算要进行修改的位数，将汉明码一致校验阵 M 与载体矩阵 a 进行矩阵乘法运算，将得到的结果记为 temp 矩阵，然后取秘密信息的 3 位，记录在矩阵 data_bit 中。然后将 temp 矩阵模 2 后与 data_bit 进行按位异或操作，将得到的结果记为 n，将 n 转为十进制表示即为要改写的位。

如果得到要改写的位为 0，则不用对隐写载体进行修改，若要改写的位 n 不为 0，则要对载体矩阵 a 的第 n 位按照 F4 的隐写规则进行改写。

在改写结束后需要判断改写后当前位置的 AC 系数是否为 0，如果改写后的结果为 0，则需要跳过该位，并且重新对取出的秘密信息进行隐写如果改写后的结果不为 0 则隐写完成，继续寻找能够隐写的载体隐写下一组秘密信息。

6.4.5 位置逆置乱

在隐写操作进行完毕后需要将位置置乱的结果进行逆操作，将 DCT 系数恢复到原来的顺序，在位置逆置乱的过程中同样采用循环的形式，经过 times 次循环对图像的 JPEG 系数进行逆置换，变换的运算算法如下所示

$$xx = mod((c_a * c_b + 1) * (x - 1) - c_b * (y - 1), N) + 1$$

$$yy = mod(-c_a * (x - 1) + (y - 1), N) + 1$$

在经过图像的逆置换后图像的 JPEG 系数的位置恢复到进行完 DCT 变换后的位置。

6.4.6 熵编码

按照 JPEG 标准对 DCT 系数调用函数 idct 进行逆 DCT 变换，然后由于原图为 png 图片，在逆 dct 变换后需要将得到的矩阵与原图的绿色蓝色部分的矩阵进行混合，然后得到的图像即为隐写之后的图像，完成对秘密信息的隐写操作。

七、 实验结果与分析:

（一）实验 1.1：LSBM 隐写算法

• LSBM 隐写结果运行 LSBM 隐写程序后会生成四个图像，其中第一列为原图和隐写后图像的对比，第二列为原图和隐写后图像的灰度直方图的对比便于分析值对效应

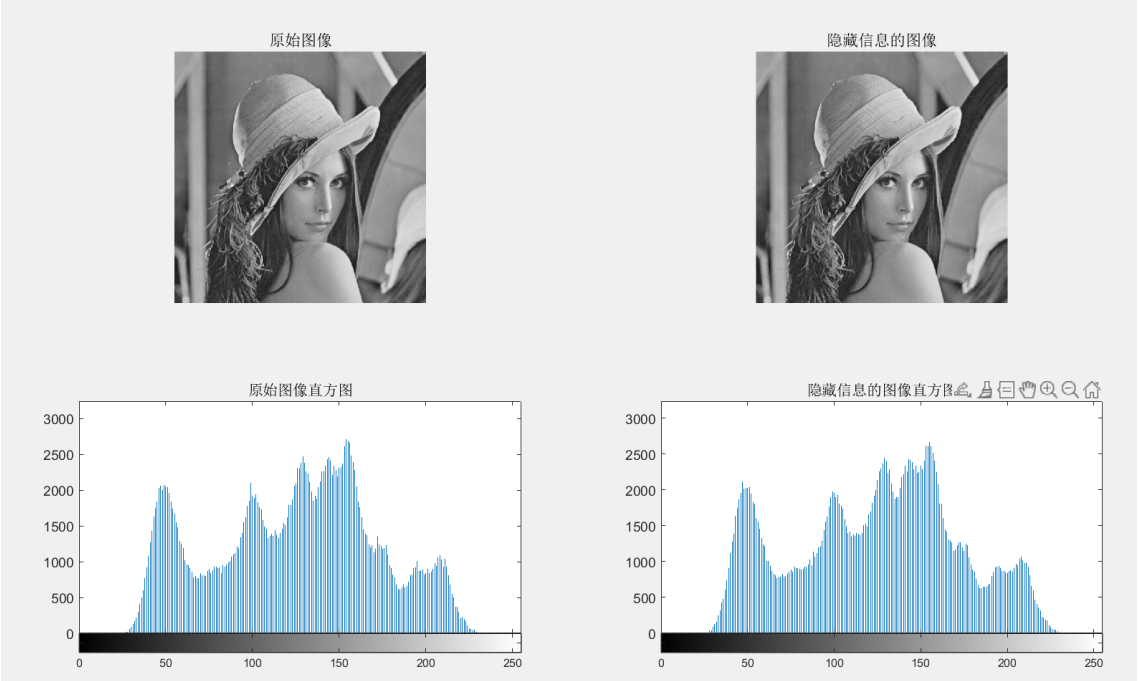

图 2: LSBM 隐写前后图像

经过观察发现当隐写率为 80% 时通过肉眼并不能看出明显的区别，观察两张图片的灰度直方图观察其值对效应发现隐写后的图像的较为不明显

• LSBM 隐写提取结果运行 LSBM 隐写提取秘密信息的程序后，得到的提取结果如下所示

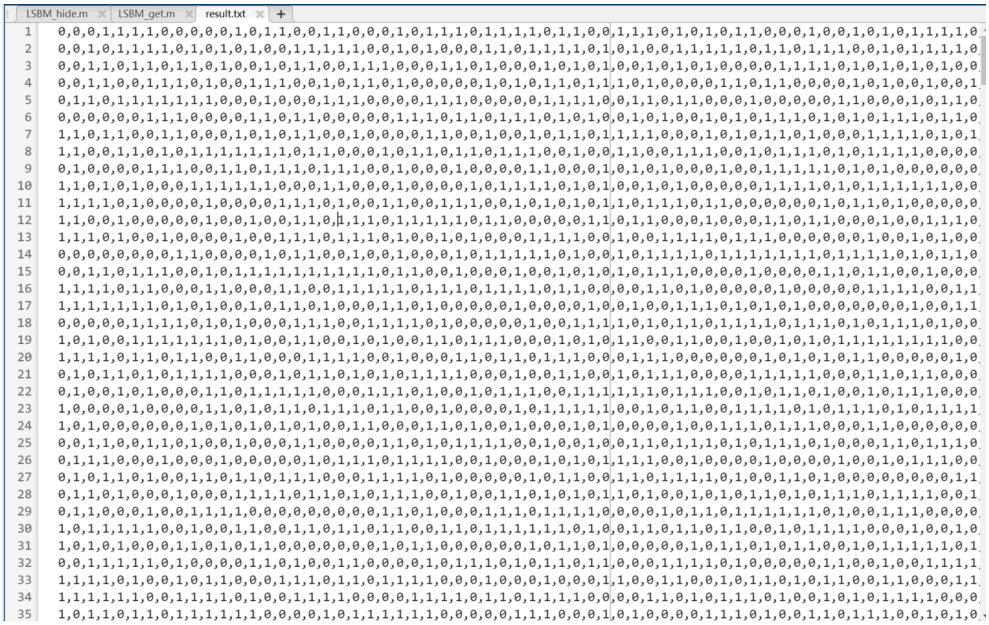

图 3: LSBM 隐写提取结果

• 对 LSBM 隐写结果进行安全性分析对 LSBM 隐写结果进行卡方分析，当嵌入率为 80% 时得到的卡方分析结果如下图所示

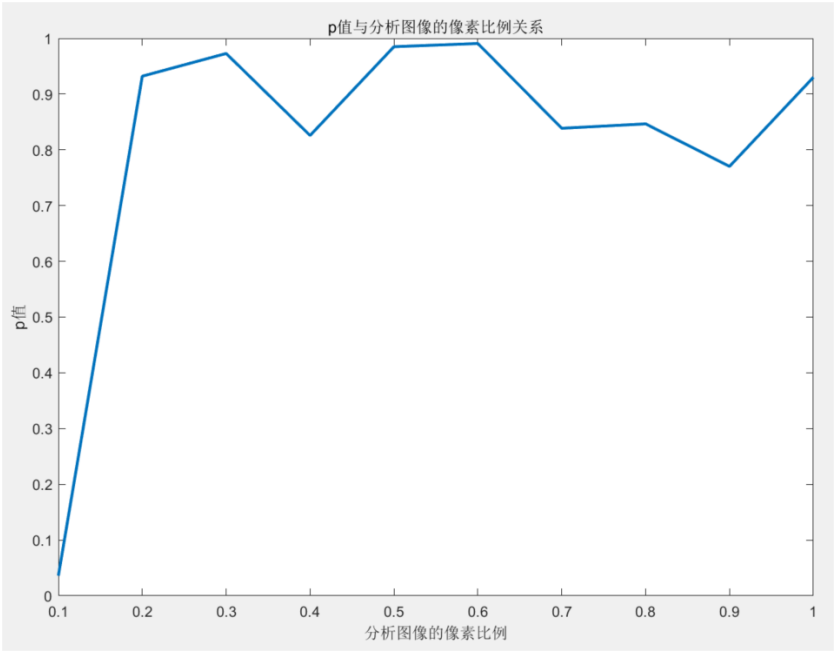

图 4: 对嵌入率为 80% 的 LSBM 隐写图像卡方分析结果

通过卡方分析结果我们可以得出当 LSBM 的嵌入率比较大达到 80% 时，卡方分析仍然不能分析出是否有信息的隐写。可以认为 LSBM 隐写方法在对抗卡方分析时是安全的

（二）实验 **1.2**：卡方分析

7.2.1 对 LSBR 进行卡方分析

• 对嵌入率为 90% 的 LSBR 隐写结果进行卡方分析当 LSBR 的嵌入率是 90% 时，得到的卡方分析结果如下图所示

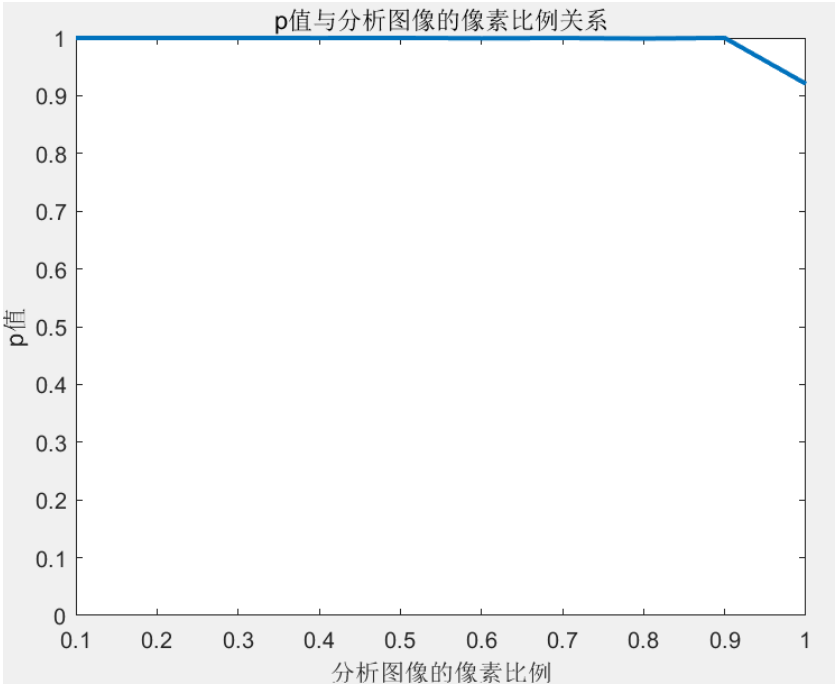

图 5: 对嵌入率为 90% 的 LSBR 隐写图像卡方分析结果

通过观察卡方分析的结果我们可以看出，p 值曲线在图像像素比例 0-0.9 的区间内保持为 1，而在 0.9-1 的区间内出现下降由此可以断定出该 LSBR 隐写的嵌入率为 90%。

• 对嵌入率为 50% 的 LSBR 隐写结果进行卡方分析当 LSBR 的嵌入率是 50% 时，得到的卡方分析结果如下图所示

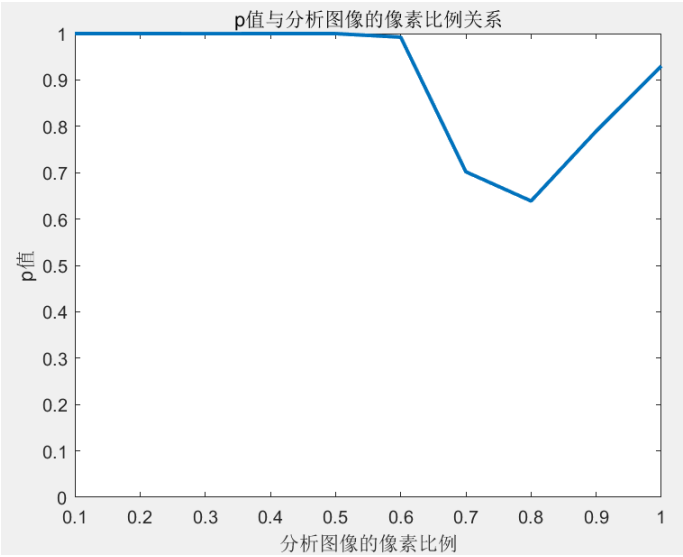

图 6: 对嵌入率为 50% 的 LSBR 隐写图像卡方分析结果

通过观察卡方分析的结果我们可以看出，p 值曲线在图像像素比例 0-0.5 的区间内保持为 1，而在 0.6-1 的区间内出现下降由此可以断定出该 LSBR 隐写的嵌入率为 50%。

• 对嵌入率为 20% 的 LSBR 隐写结果进行卡方分析当 LSBR 的嵌入率为 20% 时，得到的卡方分析结果如下图所示

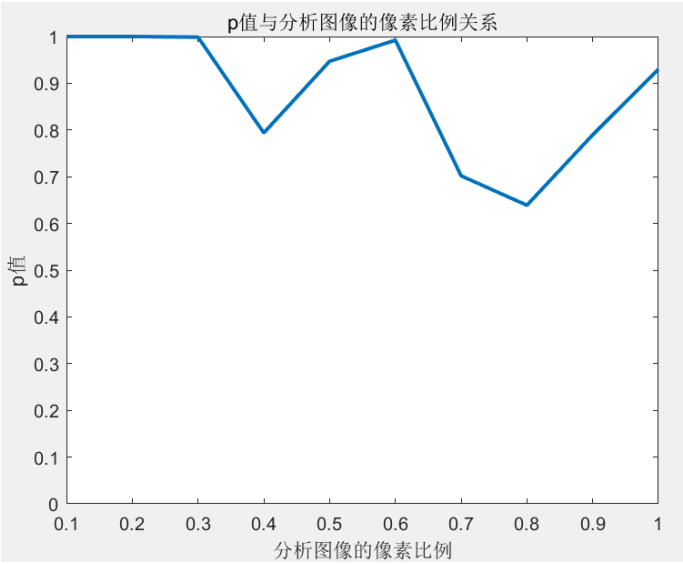

图 7: 对嵌入率为 20% 的 LSBR 隐写图像卡方分析结果

通过观察程序运行的结果我们可以看出，p 值曲线在像素比例 0-0.2 的区间内保持为 1，而在 0.3-1 的区间内出现下降由此可以断定出该 LSBR 隐写的嵌入率为 20%。

• 卡方分析对 LSBR 隐写的分析结果对 LSBR 嵌入率为 20%，50%，90% 分别进行卡方分析可以得出结果，卡方分析可以分析出伪随机 LSBR 嵌入是否隐藏信息以及隐藏信息的长度

7.2.2 对 LSBM 进行卡方分析

• 对嵌入率是 90% 的 LSBM 隐写结果进行卡方分析当 LSBM 的嵌入率是 90% 时，得到的卡方分析结果如下图所示

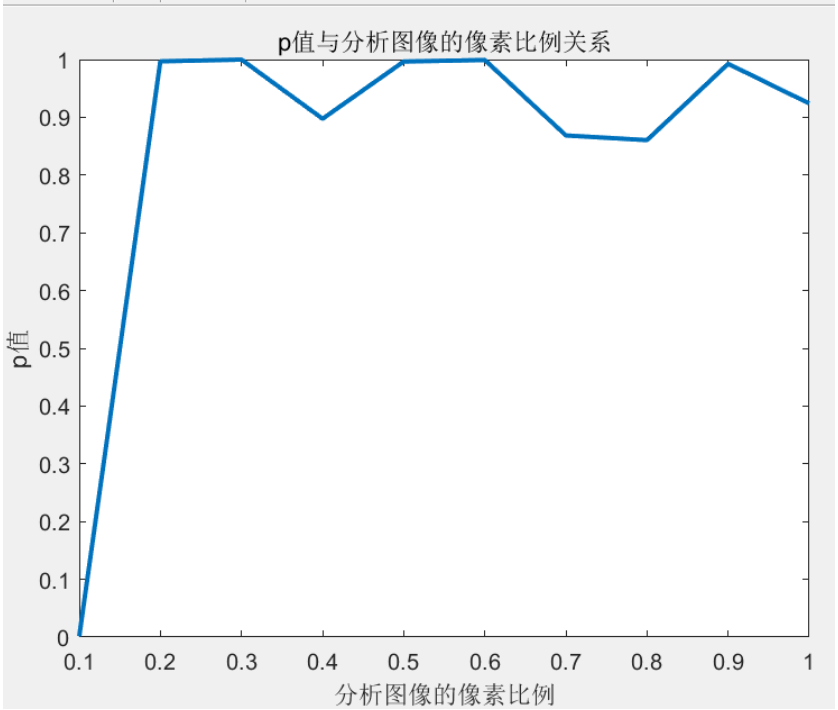

图 8: 对嵌入率为 90% 的 LSBM 隐写图像卡方分析结果

通过观察程序运行结果图像我们并不能看出 p 值曲线在像素比例下的变化趋势，并不能判断出图像是否嵌入信息以及信息的嵌入率是多少

• 对嵌入率是 50% 的 LSBM 隐写结果进行卡方分析当 LSBM 的嵌入率是 50% 时，得到的卡方分析结果如下图所示

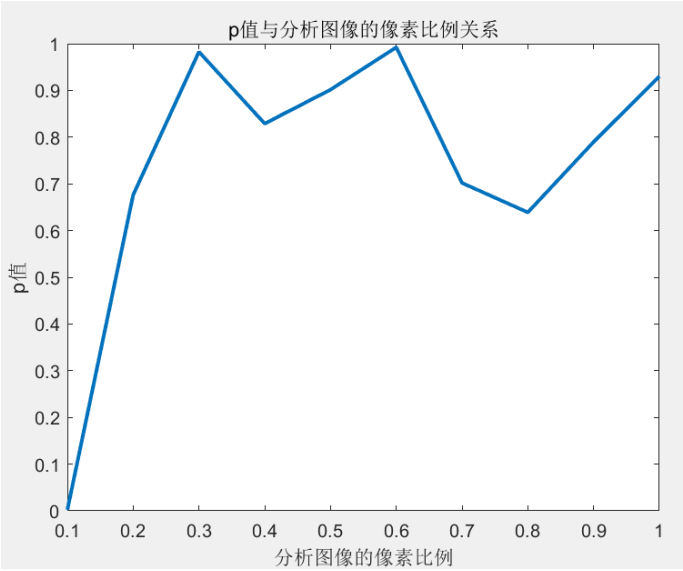

图 9: 对嵌入率为 50% 的 LSBM 隐写图像卡方分析结果

通过观察程序运行结果图像我们并不能看出 p 值曲线在像素比例下的变化趋势，并不能判断出图像是否嵌入信息以及信息的嵌入率是多少

• 对嵌入率是 20% 的 LSBM 隐写结果进行卡方分析当 LSBM 的嵌入率是 20% 时，得到的卡方分析结果如下图所示

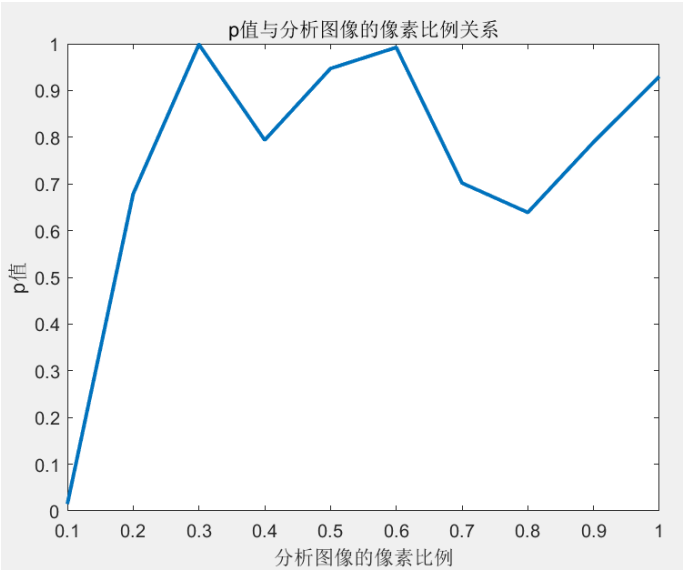

图 10: 对嵌入率为 20% 的 LSBM 隐写图像卡方分析结果

通过观察程序运行结果图像我们并不能看出 p 值曲线在像素比例下的变化趋势，并不能判断出图像是否嵌入信息以及信息的嵌入率是多少

• 卡方分析对 LSBR 隐写的分析结果对 LSBM 嵌入率为 20%，50%，90% 分别进行卡方分析可以得出结果，卡方分析不能分析出 LSBM 嵌入是否隐藏信息以及隐藏信息的长度

（三）实验 2.1：F4 隐写算法

• 对 lena 的 png 图像进行 F4 隐写得到的原图与隐写后的图像如下图所示

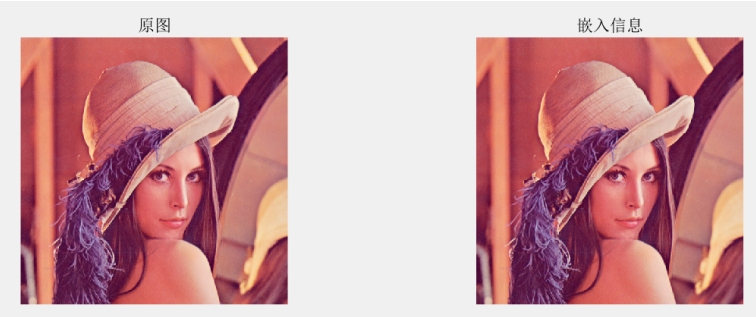

图 11: F4 隐写过后原图与隐写后图像的对比

通过观察图像可以得出 F4 隐写后的图像仅凭人眼无法识别。

• 对 lena 图像进行 F4 隐写前后得到的图像的直方图对比如下图所示

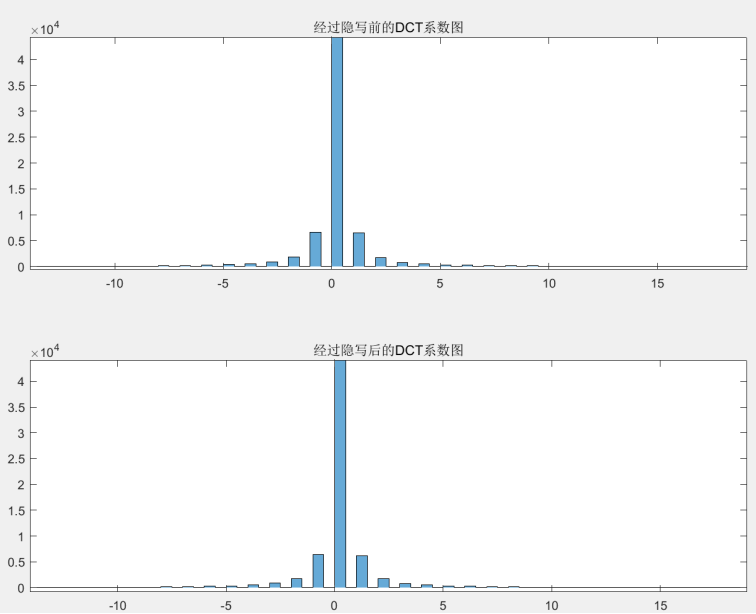

图 12: F4 隐写过后原图与隐写后图像直方图的对比

通过观察 F4 隐写后的直方图可以发现，经过 F4 隐写过后的图片并不存在值对效应，而且相比于 F3 的替换算法并不会使载密图像 DCT 系数中的偶系数增多。

• 对 F4 隐写后的图像进行信息提取得到的结果如下图所示

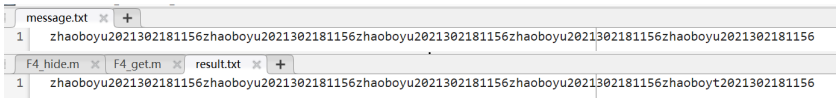

图 13: 提取 F4 隐写后的图像的结果

通过观察提取结果可以发现，该隐写方法并不能够将信息完整的提取出来，而是存在有一定的误码率。

• 将 F4 隐写后的图像进行安全性分析将 F4 隐写过后的图像输入到卡方分析程序中得到的分析结果如下图所示

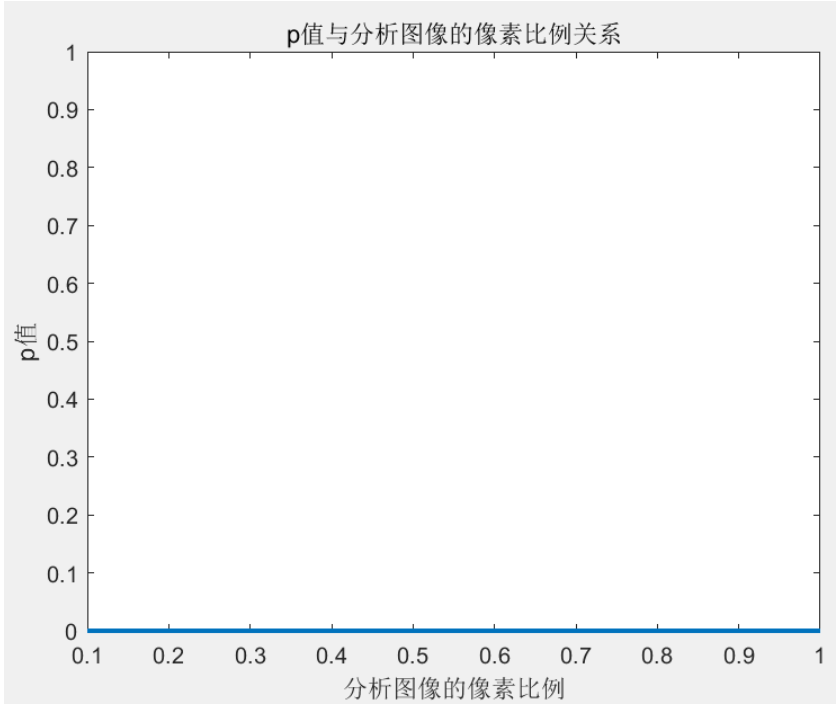

图 14: 对 F4 隐写后的结果进行卡方分析

观察卡方分析结果，p 值恒为 0 没有变化，可以得出结论 F4 隐写能够对抗卡方分析的检测，在卡方分析下具有安全性。

（四）实验 **2.2：F5** 隐写算法

- 将图像进行 F5 隐写过后生成的载密图像与原图像的对比如下图所示

图 15: 对 F5 隐写后的原图与载密图像的对比

- 对 F5 隐写后的图像进行信息提取得到的结果如下图所示

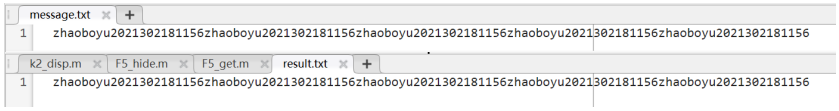

图 16: 提取 F5 隐写后的图像的结果

观察提取结果可得该隐写方法能够将隐写信息完整的提取出来

- 对 F5 隐写算法进行安全性分析对 F5 隐写算法后的载密图像进行卡方分析得到的结果如下图所示

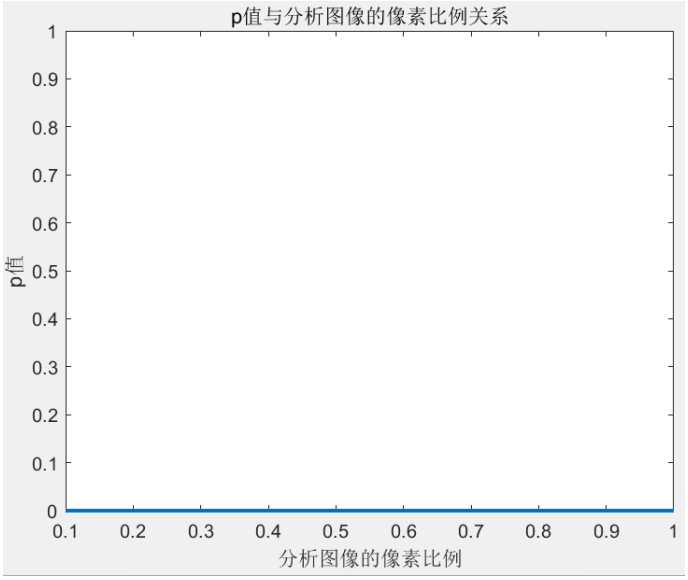

图 17: F5 隐写后的载密图像进行卡方分析

观察卡方分析结果，p 值恒为 0 没有变化，可以得出结论 F5 隐写能够对抗卡方分析的检测，在卡方分析下具有安全性。

• 对 F5 隐写结果修改的次数进行比较分析对 LSBR，LSBM，F4，F5 隐写算法中修改的数量进行统计，统计结果如下表所示

表 1: 每种隐写方案修改次数统计

隐写方法	隐写信息长度（bit）	修改次数	每次修改隐写信息长度
LSBR	235929	236032	0.9996
LSBM	52428	26109	2.0080
F4	840	1324	0.6344
F5	840	472	1.7797

上述统计结果中 LSBR 的嵌入率为 90%,LSBM 的嵌入率为 20%。

通过比较结果我们可以得出在四种隐写方法中 LSBM 每次修改能够隐写的信息的长度最多，而 F4 每次修改能够隐写的信息长度最少 F5 每次修改隐写信息长度相比较于 F4 有非常大的提高，而且其安全性也明显高于 LSBR 和 LSBM 隐写算法。

因此 F5 隐写算法能够保持在相对安全的前提下能够每次修改更多隐写信息

八、 总结及心得体会:

1. 在本次实验中学习到了 LSBM 隐写算法的操作方法以及其相对于 LSBR 隐写算法的改进效果。

2. 在本次实验中学习到了卡方分析这一检测信息隐写的方法，学会了使用其分析 LSBR 得到隐写的嵌入率的方法

3. 学习到了 F4 隐写算法的实现方法，了解到 F4 隐写算法相对于 Jsteg 隐写算法不仅消除了值对效应，相对于 F3 算法消除了隐写后偶系数变多的效应.

4. 学习到了 F5 隐写算法的实现方法，了解到通过汉明码一致校验阵进行信息的隐藏来减少每个载体分组中修改次数从而提高嵌入率的方法。

5. 体会到信息的安全隐写隐藏和隐写信息的检测与分析是相辅相成共同进步

的关系，二者在对抗中不断进行功能的完善和效率的提高

6. 体会到了任何的隐写算法都会造成载密图像与原图像之间的差别，并不存在一种完全不能被检测出来的隐写算法，但只存在有更为安全的能够对抗更多分析算法的隐写算法。