浙江大学

课程名称:	多媒体安全
姓　　名:	
学　　院:	计算机学院
专　　业:	计算机科学与技术
学　　号:	
指导老师:	黄劲
完成时间:	2023 年 6 月 21 日

实验五：F5 隐写术分析

一、实验目的

1. 了解数字图像隐写术中的典型算法 F5 的基本原理。
2. 了解矩阵编码技术的基本原理。
3. 掌握嵌入效率的计算。

二、实验内容与要求

1. 实现 F5 的隐写系统，包括信息嵌入与信息检测。
2. 设计一份 3KB 左右的文本信息数据，并使用至少两种不同的矩阵编码技术进行信息的嵌入。分析比较不同矩阵编码的嵌入效率，并绘制原图与嵌入后得到的结果图的 DCT 系数直方图。
3. 在 F5 隐写系统中，增加混洗技术，即将 DCT 系数打乱后再使用矩阵编码技术进行信息嵌入。分析混洗技术对信息隐写带来的影响。

三、实验环境

语言版本：MATLAB R2020b

四、实验过程

4.1 Jpeg 量化处理

此部分主要是实现图像转换为 DCT 系数行向量，以及从该行向量重建图像。

将图像转换为对应的 DCT 系数需要对每个图像的 8*8 小块单独进行 DCT 变换，取整后得到 DCT 量化系数。

定义函数 getJpegCoe() 由图像数值矩阵获取其 DCT 量化系数矩阵，如下：

```
function [mat] = getJpegCoe(mat)

    Quan = [
        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;
    ];
    for i = 1 : 64
        for j = 1 : 64
            part = mat((i * 8 - 7) : (i * 8), (j * 8 - 7) : (j * 8));
            part = round(dct2(part) ./ Quan);
            mat((i * 8 - 7) : (i * 8), (j * 8 - 7) : (j * 8)) = part;
        end
    end
end
```

之后，通过一个排列矩阵把所有高频系数排列在行向量的最前面，以方便隐写术植入信息的同时最小化对图像的扰动。

定义函数 rearrangeCoe() 由 DCT 量化系数矩阵获取排列后的 DCT 系数行向量，如下:

```
function [lst] = rearrangeCoe(image)

    Zigzag = [
        64, 63, 59, 58, 50, 49, 37, 36;
        62, 60, 57, 51, 48, 38, 35, 22;
        61, 56, 52, 47, 39, 34, 23, 21;
        55, 53, 46, 40, 33, 24, 20, 11;
        54, 45, 41, 32, 25, 19, 12, 10;
        44, 42, 31, 26, 18, 13, 9, 4;
        43, 30, 27, 17, 14, 8, 5, 3;
        29, 28, 16, 15, 7, 6, 2, 1;
    ];
    lst = zeros([1, 512 * 512]);
    for i = 1 : 64
        for j = 1 : 64
```

```
            for k = 1 : 8
                for l = 1 : 8
                    lst(((i - 1) * 64 + j - 1) * 64 + Zigzag(k, l)) =
                    image((i - 1) * 8 + k, (j - 1) * 8 + l);
                end
            end
        end
    end
end
```

之后再实现以上两个函数的逆变换 deJpeg() 和 restoreCoe() 即可，代码略。

4.2 F5+ 矩阵编码的隐写及检测

此部分是实现 F5 隐写术植入信息以及对信息进行 F5 检测。

在我们将要实现的 F5 系统中，应用了矩阵编码的基本原理。例如，应用矩阵

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

进行编码意为：每三个（列数）DCT 系数嵌入两个（行数）码字；第一个码字由系数 1 和系数 2 异或取最低位得到；第二个码字由系数 2 和系数 3 异或取最低位得到。

嵌入时，根据每一组系数的异或情况选择让最少的位数异或 1 即可。

定义函数 F5_embed_1() 根据上述矩阵、待植入向量 lst 和隐写信息 message 完成信息植入并返回。实现如下：

```
function [lst] = F5_embed_1(lst, message)

    i = 1;
    j = 1;
    while j <= length(message)
        b1 = mod(lst(i + 1) - lst(i), 2);
        b2 = mod(lst(i + 1) - lst(i + 2), 2);
        if b1 ~= message(j) && b2 ~= message(j + 1)
            lst(i + 1) = bitxor(cast(lst(i + 1), 'int32'), 1);
        elseif b1 ~= message(j)
            lst(i) = bitxor(cast(lst(i), 'int32'), 1);
```

```
        elseif b2 ~= message(j + 1)
            lst(i + 2) = bitxor(cast(lst(i + 2), 'int32'), 1);
        end
        i = i + 3;
        j = j + 2;
    end
end
```

定义函数 F5_decode_1() 根据上述矩阵、待解码向量 lst 和信息长度 len 完成信息解码并返回解码信息 message。实现如下:

```
function [message] = F5_decode_1(lst, len)

    message = zeros([1, len]);
    i = 1;
    j = 1;
    while j <= len
        message(j) = mod(lst(i + 1) - lst(i), 2);
        message(j + 1) = mod(lst(i + 1) - lst(i + 2), 2);
        i = i + 3;
        j = j + 2;
    end
end
```

此外，为了展示不同矩阵编码方式的效果，采用如下矩阵作为对比:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

对应的编码函数 F5_embed_2():

```
function [lst] = F5_embed_2(lst, message)

    i = 1;
    j = 1;
    while j <= length(message)
```

```
        b1 = mod(lst(i) - lst(i + 1), 2);
        b2 = mod(lst(i) - lst(i + 2), 2);
        b3 = mod(lst(i) - lst(i + 3), 2);
        if b1 ~= message(j) && b2 ~= message(j + 1) &&  b3 ~= message(j + 2)
            lst(i) = bitxor(cast(lst(i), 'int32'), 1);
        elseif b1 ~= message(j) && b2 ~= message(j + 1)
            lst(i + 1) = bitxor(cast(lst(i + 1), 'int32'), 1);
            lst(i + 2) = bitxor(cast(lst(i + 2), 'int32'), 1);
        elseif b1 ~= message(j) && b3 ~= message(j + 2)
            lst(i + 1) = bitxor(cast(lst(i + 1), 'int32'), 1);
            lst(i + 3) = bitxor(cast(lst(i + 3), 'int32'), 1);
        elseif b2 ~= message(j + 1) && b3 ~= message(j + 2)
            lst(i + 2) = bitxor(cast(lst(i + 2), 'int32'), 1);
            lst(i + 3) = bitxor(cast(lst(i + 3), 'int32'), 1);
        elseif b1 ~= message(j)
            lst(i + 1) = bitxor(cast(lst(i + 1), 'int32'), 1);
        elseif b2 ~= message(j + 1)
            lst(i + 2) = bitxor(cast(lst(i + 2), 'int32'), 1);
        elseif b3 ~= message(j + 2)
            lst(i + 3) = bitxor(cast(lst(i + 3), 'int32'), 1);
        end
        i = i + 4;
        j = j + 3;
    end
end
```

对应的解码函数 F5_decode_2():

```
function [message] = F5_decode_2(lst , len)

    message = zeros([1, len]);
    i = 1;
    j = 1;
    while j <= len
        message(j) = mod(lst(i) - lst(i + 1), 2);
```

```
            message(j + 1) = mod(lst(i) - lst(i + 2), 2);
            message(j + 2) = mod(lst(i) - lst(i + 3), 2);
            i = i + 4;
            j = j + 3;
        end
    end
```

4.3 混洗技术实现

此部分是实现 DCT 系数的打乱和复原。

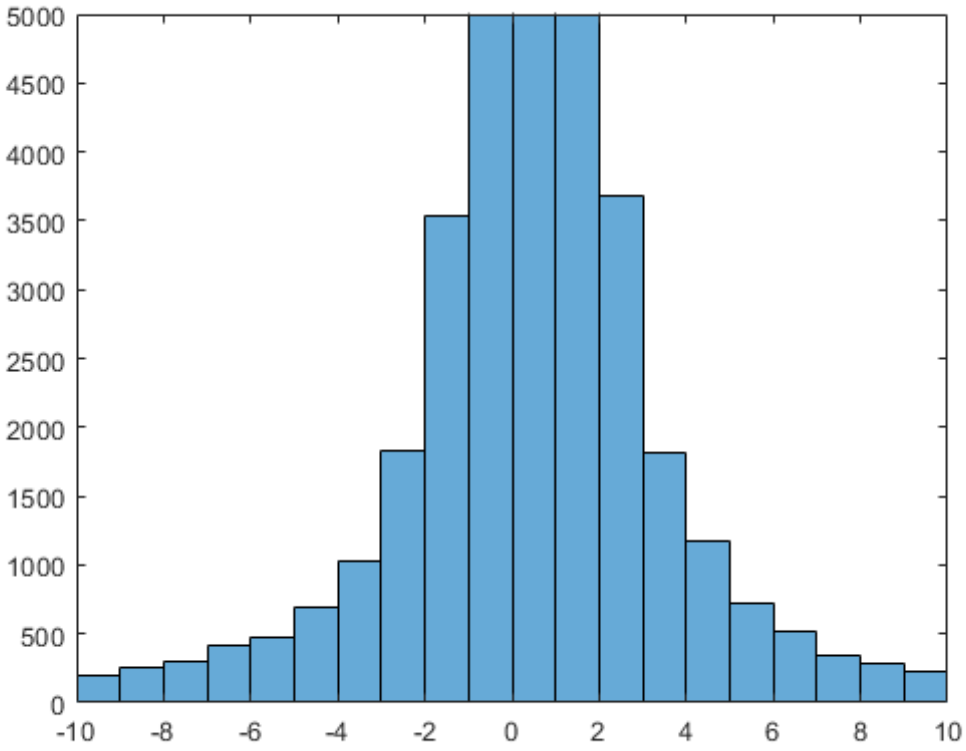
这里用到的随机化方式是，随机生成一个排列向量，然后用这个排列向量对 DCT 系数进行重排和复原。

```
perm = randperm(512 * 512);
```

4.4 F5 隐写术效果展示

原图为:

使用 histogram() 绘制原图 DCT 系数的直方图，限制 x 轴为 [-10, 10]，y 轴为 [0, 5000]（忽略 ±1 和 0 的高度），得到的直方图如下。

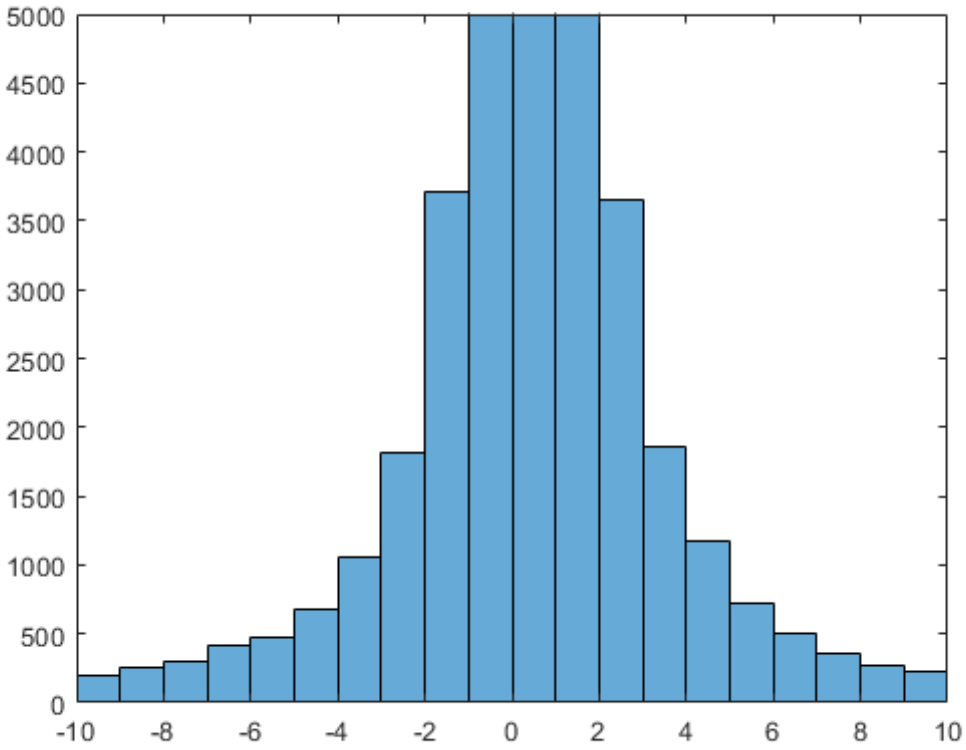


采用矩阵

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

对原图进行 F5 隐写，所得图片如下：

使用 histogram() 绘制隐写后 DCT 系数的直方图，限制 x 轴为 [-10, 10]，y 轴为 [0, 5000]（忽略 ±1 和 0 的高度），得到的直方图如下。



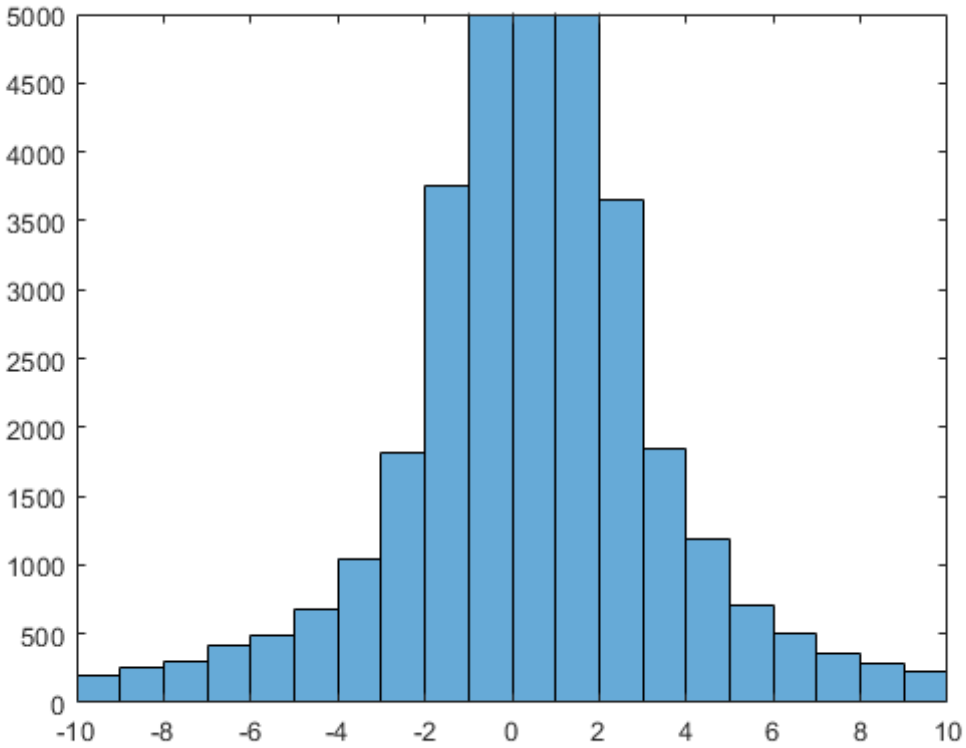
从隐写后的未保存图片读取隐写信息，准确率为 100%。

采用矩阵

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

对原图进行 F5 隐写，所得图片如下：

使用 histogram() 绘制 F3 隐写后 DCT 系数的直方图，限制 x 轴为 [-10, 10]，y 轴为 [0, 5000]（忽略 ±1 和 0 的高度），得到的直方图如下。



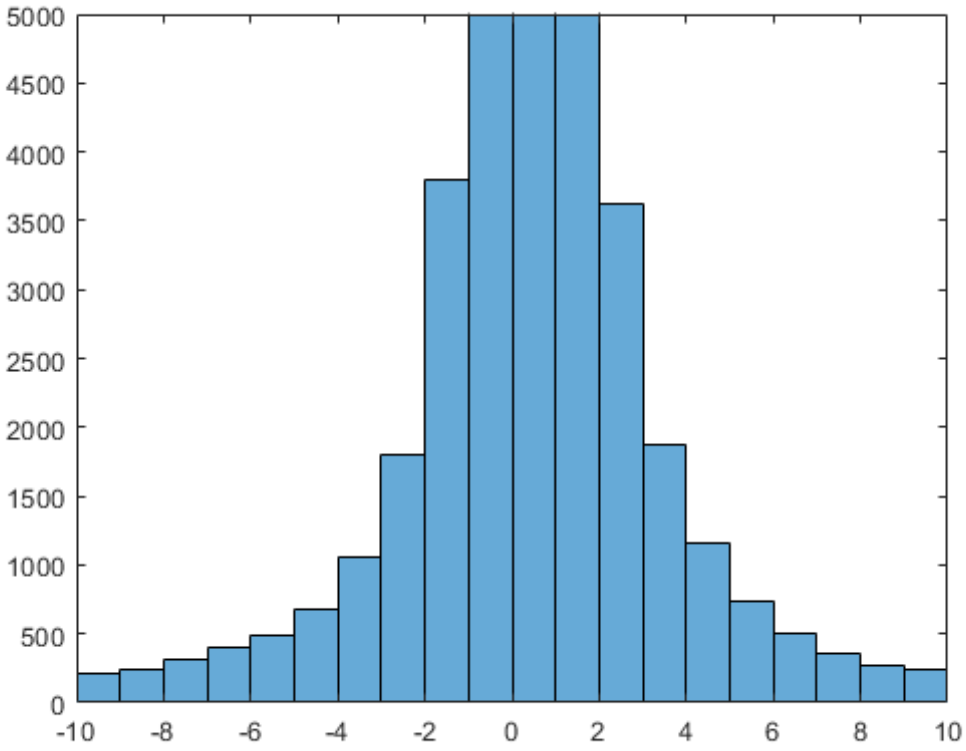
从隐写后的未保存图片读取隐写信息，准确率为 100%。

使用混洗技术，再次采用

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

进行 F5 隐写，所得图片如下：

使用 histogram() 绘制隐写后 DCT 系数的直方图，限制 x 轴为 [-10, 10]，y 轴为 [0, 5000]（忽略 ±1 和 0 的高度），得到的直方图如下。



从隐写后的未保存图片读取隐写信息，准确率为 100%。

五、实验分析与结论

5.1 F5 隐写对图像及直方图的影响

我们实现的 F5 隐写没有考虑 0 和 1 的处理，仅仅是对每一组 DCT 进行比对和异或操作，因此，在获得的直方图中 1 的计数比-1 高很多，有一定异常。不过这可以通过特殊处理 0 和 1 简单处理（但处理的话就无法嵌入 3KB 数据，故本实验中没有处理）。

其他系数直方图几乎未见异常，可见 F5 有较好的信息隐藏效果。

同时，放大图像后也可看出明显的 8*8 分块。

如果不采用混洗技术，在图像的上部可以看出明显的噪声。这是许多修改集中在前几块的 DCT 系数的结果，如下。

因此，应该采取混洗技术将这些修改尽可能随机到图像的不同块中。

5.2 混洗技术改进

采用混洗技术后，DCT 系数修改被尽可能随机到不同的块中，因此图像也不再有非常集中的明显噪音，只有较小的噪音，能够进行更好的信息隐藏。

5.3 嵌入效率分析

使用矩阵

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

时，4 种编码情况有 3 种需要修改 1bit，有 1 种不需要修改，故嵌入效率：

$$\frac{2}{1 * \frac{3}{4} + 0 * \frac{1}{4}} = \frac{8}{3}$$

。

使用矩阵

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

时，8 种编码情况有 3 种需要修改 2bit，有 4 种需要修改 1bit，有 1 种不需要修改，故嵌入效率：

$$\frac{3}{2 * \frac{3}{8} + 1 * \frac{4}{8} + 0 * \frac{1}{8}} = \frac{12}{5}$$

。

可见，矩阵编码的嵌入效率受矩阵选取影响较大。选择好的编码矩阵能提高嵌入效率，也能通过低修改位数提高容错性和鲁棒性。

六、实验感想

本次实验在做过 F3 和 F4 的基础上做起来比较简单，没有什么问题。虽然说植入 3KB 确实对隐写后图片影响太大了，不过也是便于分析。

通过本实验，我理解并掌握了 F5 隐写术的基本方法及原理，加强了我对隐写术和隐写算法的理解。