**信息隐藏与数字水印——变换域的图像数字水印**

57119101 王晨阳

**目录**

[1实验目的 2](#_Toc106963966)

[2实验原理 2](#_Toc106963967)

[3实验流程 2](#_Toc106963968)

[4实验结果 5](#_Toc106963969)

[3.1直接输出 5](#_Toc106963970)

[3.2修改嵌入深度 6](#_Toc106963971)

[3.3 添加高斯白噪声 6](#_Toc106963972)

[3.4 经过高斯低通滤波 7](#_Toc106963973)

[3.5 经过JPEG压缩 9](#_Toc106963974)

[3.6 裁剪 10](#_Toc106963975)

[3.7 旋转 11](#_Toc106963976)

[3.8 改进 13](#_Toc106963977)

[5 实验结论 14](#_Toc106963978)

[6实验心得 14](#_Toc106963979)

[附录 15](#_Toc106963980)

# 1实验目的

任务是实现对 640×480 的灰度图像（样本自选，可以是 lena 图像）进行数字水印设计。水印本体选择用东南大学校徽图像，大小自定 。应用变换域数字水印方法，例如基于 DCT 方法等进行实验并测试 。对上述技术和方法进行实验 、 结果分析等，特别是对其的健壮性、抗攻击进行分析 。

# 2实验原理

DCT，即Discrete Cosine Transform，意为离散余弦变换，其常见用途是对音视频进行数据压缩。DCT 将原始图像信息块转换成代表不同频率分量的系数集，这有两个优点：其一，信号常将其能量的大部分集中于频率域的一个小范围内，这样一来，描述不重要的分量 只需要很少的比特数；其二，频率域分解映射了人类视觉系统的处理过程，并允许后继的量化过程满足其灵敏度的要求。

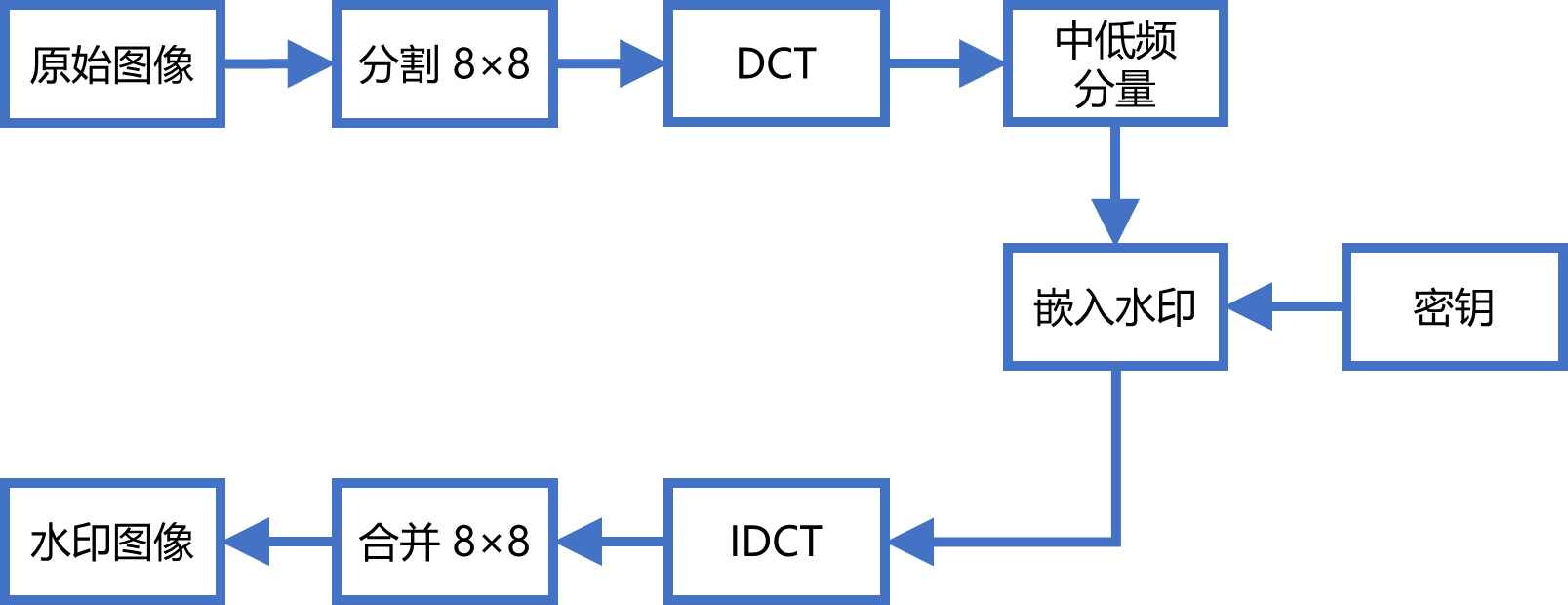
对于一维DCT，其计算方法如公式（1）所示。

而对于二维DCT，其计算方法如公式（2）所示。

由于频域信息对应的是图像的“清晰度”，即高频部分为图像较为锐利的部分，故我们可以利用中低频部分来隐藏信息而不被察觉。但是，在图像压缩算法中，往往会滤去低频部分以达到较好的压缩效果，故我们选择中频来隐藏信息。

# 3实验流程

信息隐藏的步骤如图1所示。



**图1** 信息隐藏流程图

嵌入时，我们首先将图像分割成 8×8 的子区域，然后对每个区域分别做 DCT，得到的结果在中频分量中嵌入水印，然后再还原为图像。核心代码如下：

x=(i-1)\*8;   
y=(j-1)\*8;   
if watermark\_img(i,j)==1   
    k=k1;   
else   
    k=k2;   
end   
after(x+1,y+8)=before(x+1,y+8)+alpha\*k(1);   
after(x+2,y+7)=before(x+2,y+7)+alpha\*k(2);   
after(x+3,y+6)=before(x+3,y+6)+alpha\*k(3);   
after(x+4,y+5)=before(x+4,y+5)+alpha\*k(4);   
after(x+5,y+4)=before(x+5,y+4)+alpha\*k(5);   
after(x+6,y+3)=before(x+6,y+3)+alpha\*k(6);   
after(x+7,y+2)=before(x+7,y+2)+alpha\*k(7);   
after(x+8,y+1)=before(x+8,y+1)+alpha\*k(8);

我们首先生成两个随机序列 k1 和 k2 作为密钥。嵌入时，如果像素位为 1，则嵌入 k1，否则嵌入 k2。我们选择了 (x+1,y+8)、(x+2,y+7)、(x+3,y+6)、(x+4,y+5)、(x+5,y+4)、(x+6,y+3)、(x+7,y+2)、(x+8,y+1) 共 8 个位置嵌入，以增强水印鲁棒性。alpha 为嵌入深度，其越大，则鲁棒性越强，但水印也更容易被发现。

提取时，我们同样首先将图像分割成 8×8 的子区域，然后对每个区域分别做 DCT，得到的结果在中频分量中提取水印，然后再还原为图像。核心代码如下：

x=(i-1)\*8;   
y=(j-1)\*8;   
after(1)=before(x+1,y+8);   
after(2)=before(x+2,y+7);   
after(3)=before(x+3,y+6);   
after(4)=before(x+4,y+5);   
after(5)=before(x+5,y+4);   
after(6)=before(x+6,y+3);   
after(7)=before(x+7,y+2);   
after(8)=before(x+8,y+1);   
if corr2(after,k1)>corr2(after,k2)   
    watermark\_img(i,j)=1;   
else   
    watermark\_img(i,j)=0;   
end

我们计算了提取得到的内容分别与 k1、k2 的相关性，如果 k1 相关性更大，则提取出 1，反之亦然。

我们选取的载体图片为 lena，水印图片为东南大学校徽。具体如下所示：



**图2** lena 640×480



**图3** 东南大学校徽60×60

其中，lena 为 640×480，校徽图片为 60×60，这是因为载体图片的宽度限制了水印图片的最大宽度为

在实现水印的嵌入与提取后，我们还对其鲁棒性进行了测试。我们分别对嵌入深度、高斯白噪声大小、高斯低通滤波标准差、JPEG 压缩程度、裁剪、旋转等情况进行了研究。

# 4实验结果

## 3.1直接输出

我们首先测试了水印的嵌入与直接输出，输出结果如下：



**图4** lena输出

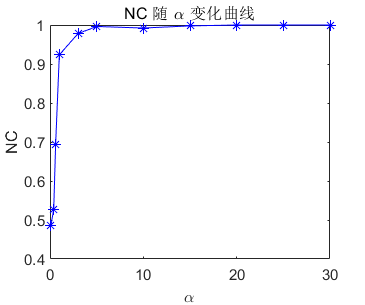


**图5** 东南大学校徽输出

可以看到，输出得到的水印与嵌入时一模一样。

## 3.2修改嵌入深度

我们修改 alpha 的值，并计算归一化相关系数NC，以评价水印嵌入的鲁棒性。我们得到结果如图6所示。



**图6** NC 随 变化曲线

可以看出，随着嵌入深度增加，NC 急剧增加，并在 时达到最高值附近。

## 3.3 添加高斯白噪声

我们对嵌入水印后的图像加入高斯白噪声。在白噪声为 0.1dB 的情况下，输出如下所示。

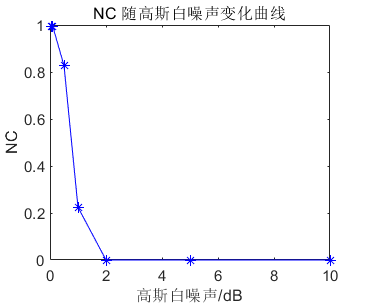


**图7** 高斯白噪声下lena输出



**图8** 高斯白噪声下东南大学校徽输出

我们修改白噪声大小，作出了以下曲线。



**图9** 高斯白噪声下NC 随白噪声大小变化曲线

可以看到，噪声越大，NC越小，在噪声达到2dB时，提取出的水印已经完全不可辨认。

## 3.4 经过高斯低通滤波

我们对嵌入水印后的图像通过高斯低通滤波滤去低频成分。在滤波标准差为1的情况下，输出如下所示。

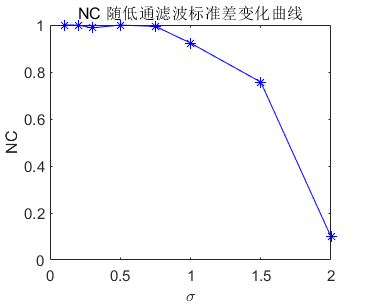


**图10** 低通滤波下lena输出



**图11** 低通滤波东南大学校徽输出

我们修改低通滤波标准差的大小，作出了以下曲线。



**图12** 低通滤波下NC 随低通滤波标准差变化曲线

可以看到，在小于0.8时，水印可以几乎完美地提取出来，而在之后，提取质量会加速下降。

## 3.5 经过JPEG压缩

我们对嵌入水印后的图像使用JPEG格式进行压缩。在图像质量为50的情况下，输出如下所示。

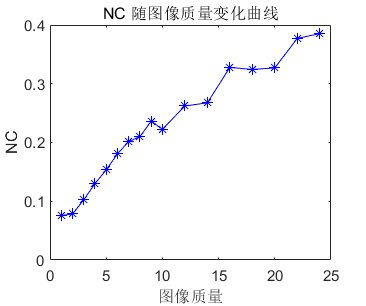


**图13** JPEG压缩下lena输出



**图14** JPEG压缩下东南大学校徽输出

我们修改图片质量的大小，作出s了以下曲线。



**图15** JPEG压缩下NC 随图像质量变化曲线

可以看到，图片质量越高，提取出的水印质量也越高，这与我们的常识是相符的。

## 3.6 裁剪

我们对嵌入水印后的图像裁剪掉一部份区域。我们裁剪掉左上角的区域，输出如下所示。



**图16** 裁剪下lena输出



**图17** 裁剪下东南大学校徽输出

可以看到，我们在裁切掉左上角后，提取出的水印左上角也相应的被裁切掉了。而如果我们裁切下侧，由于水印只利用了载体图像的上半部分，故提取出的水印不会产生影响。

## 3.7 旋转

我们对嵌入水印后的图像进行旋转。我们逆时针旋转10°，输出如下所示。



**图18** 旋转10°下lena输出



**图19** 旋转10°下东南大学校徽输出

可以看到，我们在进行旋转后，提取出的水印已经完全不可辨认。这是因为我们嵌入时将图像进行了分块，在旋转后，分块得到的内容已经和原来不同。

但是，如果我们将其旋转180°，输出如下所示。



**图20** 旋转180°下lena输出



**图21** 旋转180°下东南大学校徽输出

可以看到，可以输出原水印的下半部分，且颜色与原来相反。

## 3.8 改进

我们之前的方法中只利用了载体图像的上半部分。我们希望能够物尽其用，用完整张图片。我们将校徽图像reshape为一个向量，然后依次嵌入。640×480的载体图像最多可以嵌入

个像素。相应的，校徽图像最大边长为

我们重新生成了69×69的校徽图像，如下图所示。



**图22** 东南大学校徽69×69

嵌入后再提取水印，得到如下图像。



**图23** 东南大学校徽输出

可以看到，依然能够正常提取。

# 5 实验结论

通过以上实验，我们发现通过DCT实现图像数字水印隐藏式可行的。隐藏得到的图像可以抵抗一定范围内的噪声、低通滤波、压缩、裁剪等攻击，但旋转会使得水印无法提取。最后，我们讨论了图像可以嵌入的最大水印，并进行了实验。

# 6实验心得

通过本次实验，进一步加深了对数字水印的理解，通过matlab仿真的方式，了解了其实际应用。同时，通过对水印鲁棒性的验证，了解了图像水印攻击的方式与效果，具有极高的现实意义。

# 附录

matlab 代码（以直接提取为例）

%%

clear;

clc;

close all;

%%

watermark\_img\_original = imread('seu\_60\_60.png');

watermark\_img\_original = rgb2gray(watermark\_img\_original);

watermark\_img\_original = im2bw(watermark\_img\_original);

[row, column] = size(watermark\_img\_original);

%%

carrier\_img\_original = imread('lena\_640\_480.bmp');

carrier\_img\_original = im2gray(carrier\_img\_original);

%%

before = blkproc(carrier\_img\_original, [8 8], 'dct2');

%%

alpha = 30;

k1 = randn(1, 8);

k2 = randn(1, 8);

after = before;

for i = 1:row

    for j = 1:column

        x = (i - 1) \* 8;

        y = (j - 1) \* 8;

        if watermark\_img\_original(i, j) == 1

            k = k1;

        else

            k = k2;

        end

        after(x + 1, y + 8) = before(x + 1, y + 8) + alpha \* k(1);

        after(x + 2, y + 7) = before(x + 2, y + 7) + alpha \* k(2);

        after(x + 3, y + 6) = before(x + 3, y + 6) + alpha \* k(3);

        after(x + 4, y + 5) = before(x + 4, y + 5) + alpha \* k(4);

        after(x + 5, y + 4) = before(x + 5, y + 4) + alpha \* k(5);

        after(x + 6, y + 3) = before(x + 6, y + 3) + alpha \* k(6);

        after(x + 7, y + 2) = before(x + 7, y + 2) + alpha \* k(7);

        after(x + 8, y + 1) = before(x + 8, y + 1) + alpha \* k(8);

    end

end

result = blkproc(after, [8 8], 'idct2');

result = uint8(result);

%%

carrier\_img\_embeded = result;

%%

before = blkproc(carrier\_img\_embeded, [8, 8], 'dct2');

after = zeros(1, 8);

watermark\_img\_extract = zeros(row, column);

for i = 1:row

    for j = 1:column

        x = (i - 1) \* 8;

        y = (j - 1) \* 8;

        after(1) = before(x + 1, y + 8);

        after(2) = before(x + 2, y + 7);

        after(3) = before(x + 3, y + 6);

        after(4) = before(x + 4, y + 5);

        after(5) = before(x + 5, y + 4);

        after(6) = before(x + 6, y + 3);

        after(7) = before(x + 7, y + 2);

        after(8) = before(x + 8, y + 1);

        if corr2(after, k1) > corr2(after, k2)

            watermark\_img\_extract(i, j) = 1;

        else

            watermark\_img\_extract(i, j) = 0;

        end

    end

end

imwrite(watermark\_img\_extract, 'seu\_extract.png');

watermark\_img\_extract = uint8(watermark\_img\_extract);

NC = correlation(watermark\_img\_extract, watermark\_img\_original);

disp(NC);