**集美大学计算机工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称： 数字信号与图像处理** | **班级：计算2114** | **实验成绩：** |
| **实验编号：003** | **姓名： 庄佳强** | **实验日期：12.1** |
| **实验名称：**图像处理基础与图像变换 | **学号：202121331104** | **实验地点：陆大206** |

1. **目的**（本次实验所涉及并要求掌握的知识点）

通过本实验加深对数字图像的理解，熟悉MATLAB中的有关函数；应用DCT对图像进行变换；熟悉图像常见的统计指标，实现图像几何变换的基本方法。

1. **实验内容与设计思想**（设计思路、主要代码结构、主要
2. 选择两幅图像，读入图像并显示，同时使用Matlab计算图像的大小、灰度平均值、协方差矩阵、灰度标准差和相关系数。

使用imread读取图片。

size求图片大小。

Rgb2gray转灰度图。

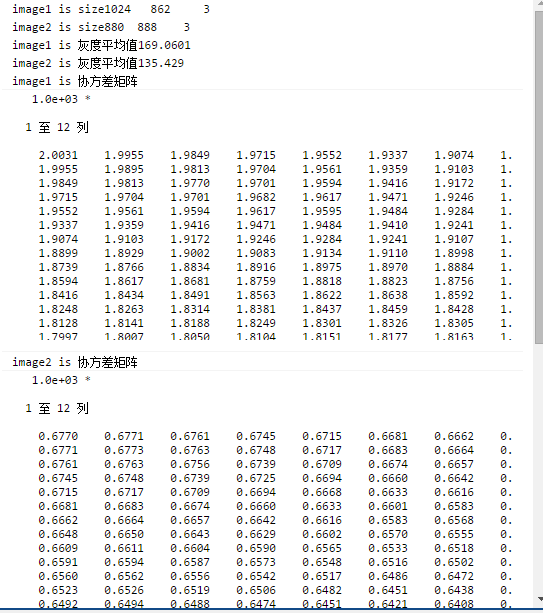
Mean2()求灰度图的灰度平均值。

Cov(double())求协方差矩阵。

Medfilt2()求中值滤波。

Std2()求标准差。

最后输出一下图片和对应灰度图。





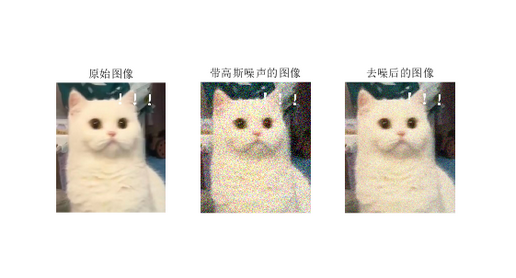
1. 图像加法去除噪声

使用for循环来实现多次操作。

I = imread('lena.tif'); %

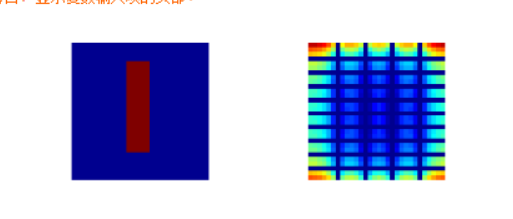
J = imnoise(I,'gaussian',0,0.01);

之后求出平均值。



**加法去噪的原理：**通过多次获取带噪声的图像，噪声是随机的，但信号是一致的。当这些图像叠加时，信号得到强化，而噪声在统计上相互抵消，因此平均后的图像中信号相对较强，而噪声相对减小，从而达到一定的去噪效果。

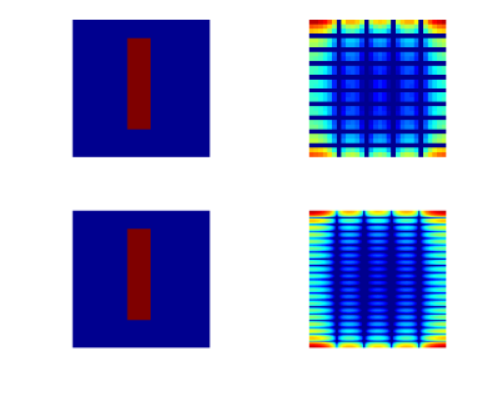
1. FFT变换



①运行程序，观察实验结果。

代码构造创建了一个具有垂直条纹的二值图像，然后进行了傅里叶变换，并显示了傅里叶变换的幅度谱。傅里叶变换的幅度谱图中亮的区域表示原始图像中的频率成分。

②修改之后观察实验结果,写出两个结果的差别



这次修改将傅里叶变换的输入尺寸扩展为256x256，然后使用fft2将零频率分量移到图像中心。这个修改在显示幅度谱时考虑了图像的频率分布。

比较与原图更加的密集和蓝色的部分尺度更加的具体。

③改变f的大小和f中矩阵的大小，说明观察结果和原来不同的原因。



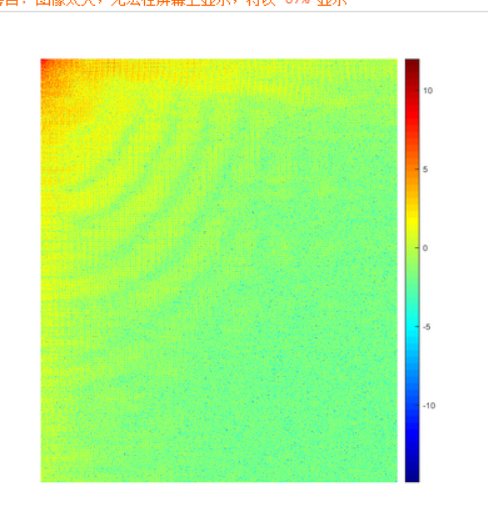
其中修改了f(10:24,13:20)=1;

观察到图像更加的扁宽，傅里叶变化幅度谱频率更大。

原因:改变了f的大小，较大的f包含更高的频率成分。

1. DCT变换

观察实验结果，说明figure2中图像色彩变换的规律，并写出其原因。



在 Figure 2 中，显示了 DCT 的幅度谱，使用了对数尺度以增强可视化效果。

原因：DCT将图像分解为一组基础函数J = dct2(I);，这些基础函数是余弦函数的变体。在图像的 DCT 幅度谱中，较亮的区域对应于具有较大幅度的 DCT 系数，而较暗的区域对应于较小幅度的 DCT 系数。

1. DCT与图像压缩
2. 运行代码，观察实验结果，写出图像压缩的基本原理。



看到了压缩之后图像中的很多特征细节都消失，图像变得更加的圆润。

原理：

DCT将图像分解为频域的系数，这些系数表示了图像中不同频率的信息。通过设置掩码，可以选择保留哪些频率分量，而将其他分量设为零，从而实现对图像的压缩。压缩后的图像在视觉上可能会失去一些细节，但可以用更少的信息来表示图像，实现了数据的压缩。

1. 计算原图像和压缩后图像的差（相减）



在差异图像中，亮的区域表示原始图像中对应位置的像素值较大，暗的区域表示对应位置的像素值较小。

1. 修改mask矩阵中非0元素的个数，写出修改后的压缩的图像和原图像之间的差和非0元素个数的关系，并比较变换后系数的重要性

修改mark为：

mask=[1 1 1 1 1 1 1 1

1 1 1 0 0 0 0 0

1 1 0 0 0 0 0 0

1 0 0 1 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1];

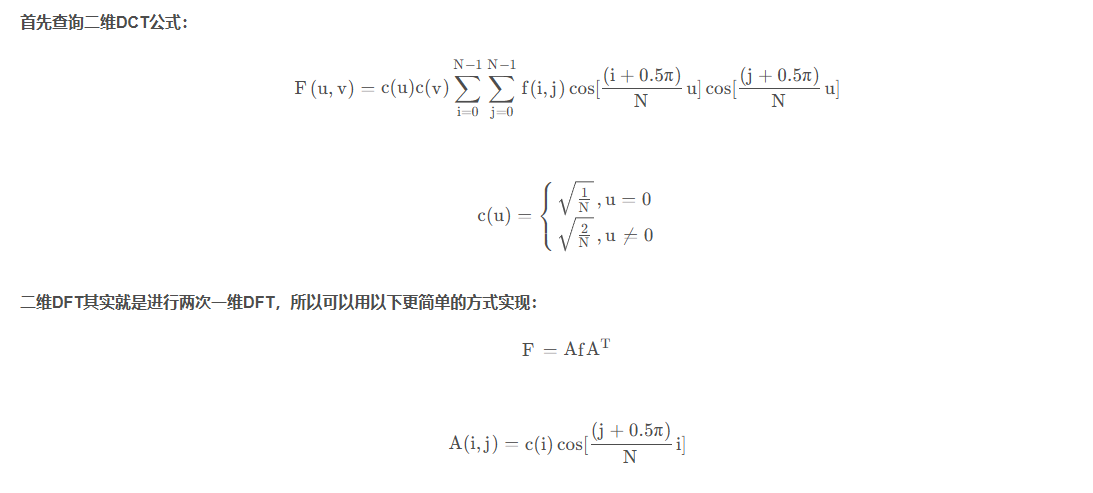


你可以观察到，随着非零元素个数的增多，压缩后的图像会少失去很多细节，而差异图像中的亮度区域会减少。同理，随着非零元素个数的减少，压缩后的图像会失去更多的细节，而差异图像中的亮度区域会增加，表示信息的丢失。

1. 使用别的图像进行上述实验，验证结论正确与否

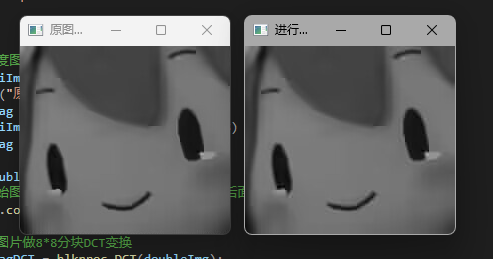


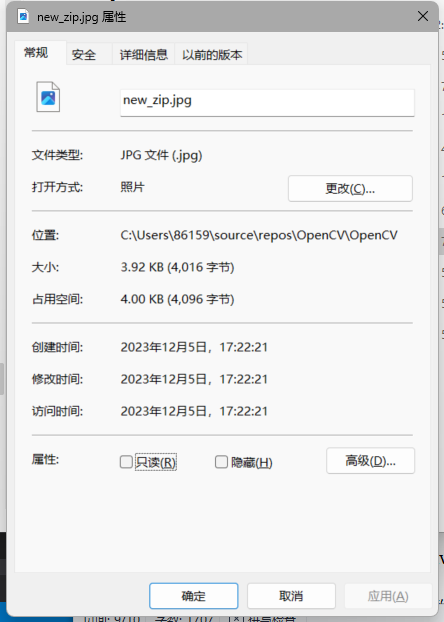
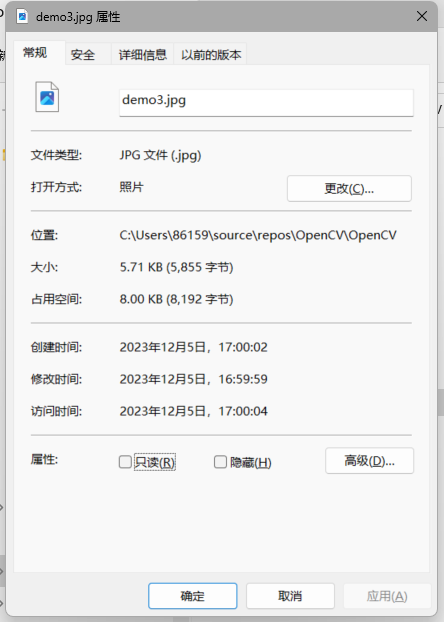
1. 使用C++与OpenCV实现（5）



N对应二维矩阵的总行数，i对应二维矩阵此时的行，j对应二维矩阵此时的列，对原图像分出的每一个块，执行公式3

得出的结果：





可以看出，缺少了一些细节部分，但是图像的整体大小缩小了。

（7）编写程序实现图像的水平垂直和中心对称（注:不能用系统的旋转函数）

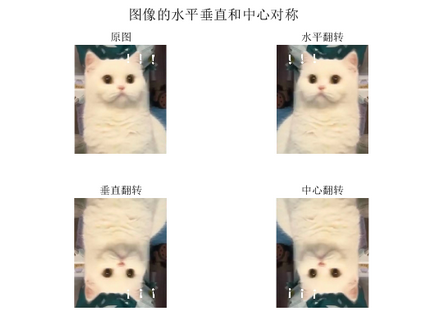
% 水平翻转

图片的数组为hwc 排布为H:W:C

：，end：-1：1，：表示宽方向中从尾开始从最后一列到第一列

对原图像进行数组的值调换，就可以实现翻转

就可以得到水平翻转的图像，其他同理哦。



**三、实验使用环境**（本次实验所使用的平台和相关软件）

Matlab,C++。

**四、参考文献**（书籍、网址链接等）

无。

1. **源代码附件**（完整的程序源代码，**注意排版紧凑，如果代码较长，字体、字间距选择小一点的**）

image1=imread('demo.jpg');

image2=imread('demo2.jpg');

%求尺寸

size1=size(image1);

disp(['image1 is size' num2str(size1)]);

size2=size(image2);

disp(['image2 is size' num2str(size2)]);

%转化为灰度

gray\_image1 = rgb2gray(image1);

gray\_image2= rgb2gray(image2);

%计算灰度平均值s

aveGrayI1 = mean2(gray\_image1);

disp(['image1 is 灰度平均值' num2str(aveGrayI1)]);

aveGrayI2 = mean2(gray\_image2);

disp(['image2 is 灰度平均值' num2str(aveGrayI2)]);

%计算协方差矩阵

%data\_vector1 = double(reshape(gray\_image1, [], 1));

cov1 = cov(double (gray\_image1));

disp('image1 is 协方差矩阵');

disp(cov1);

%data\_vector2 = double(reshape(gray\_image2, [], 1));

cov2 = cov(double (gray\_image2));

disp('image2 is 协方差矩阵');

disp(cov2);

%相关系数

I1= medfilt2(gray\_image1); % 中值滤波

r1 = corr2(gray\_image1, I1);

disp(['image1 is 中值滤波' num2str(r1)]);

I2= medfilt2(gray\_image2); % 中值滤波

r2 = corr2(gray\_image2, I2);

disp(['image2 is 中值滤波' num2str(r2)]);

%灰度标准差’

v1 = std2(gray\_image1);

v2 = std2(gray\_image2);

subplot(2,2,1);

imshow(image1);

subplot(2,2,2);

imshow(image2);

subplot(2,2,3);

imshow(gray\_image1);

subplot(2,2,4);

imshow(gray\_image2);

% 读取图像

I = imread('demo.jpg');

% 添加高斯噪声，均值为0，方差为0.01

J = imnoise(I, 'gaussian', 0, 0.01);

% 重复操作多次

num\_iterations = 5; % 可根据需要设置重复次数

summed\_images = zeros(size(I)); % 用于累加图像

for i = 1:num\_iterations

% 重新生成带噪声的 Lena 图像

J = imnoise(I, 'gaussian', 0, 0.01);

% 将生成的图像叠加到累加图像中

summed\_images = summed\_images + double(J);

end

% 取平均得到去噪后的图像

average\_image = uint8(summed\_images / num\_iterations);

% 显示原始 Lena 图像、带噪声的 Lena 图像和去噪后的 Lena 图像

figure;

subplot(1, 3, 1), imshow(I), title('原始图像');

subplot(1, 3, 2), imshow(J), title('带高斯噪声的图像');

subplot(1, 3, 3), imshow(average\_image), title('去噪后的图像');

f=zeros(30,30);f(5:24,13:17)=1;

subplot(3,2,1);imshow(f);

F=fft2(f);F2=log(abs(F));

subplot(3,2,2);imshow(F2,[-1 5]);

colormap(jet);

f=zeros(30,30);f(5:24,13:17)=1;

subplot(3,2,3);imshow(f);

F=fft2(f,256,256);F2=log(F);

subplot(3,2,4);imshow(F2,[-1 5]);

colormap(jet);

f=zeros(30,30);f(10:24,13:20)=1;

subplot(3,2,5);imshow(f);

F=fft2(f);F2=log(abs(F));

subplot(3,2,6);imshow(F2,[-1 5]);

colormap(jet);

I=imread('demo2.jpg');

I=rgb2gray(I);

I=im2double(I);

T=dctmtx(8);

B=blkproc(I,[8 8],'P1\*x\*P2',T,T');

mask=[1 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2=blkproc(B,[8 8],'P1.\*x',mask);

I2=blkproc(B2,[8 8],'P1\*x\*P2',T',T);

figure(1);

imshow(I);

figure(2),imshow(I2);

sub=I-I2;

disp('相减后');

figure(3),imshow(sub);

mask=[1 1 1 1 1 1 1 1

1 1 1 0 0 0 0 0

1 1 0 0 0 0 0 0

1 0 0 1 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1];

B3=blkproc(B,[8 8],'P1.\*x',mask);

I3=blkproc(B2,[8 8],'P1\*x\*P2',T',T);

figure(4),imshow(I3);

sub=I-I3;

figure(5);

imshow(sub);

RGB = imread('demo.jpg');

I = rgb2gray(RGB);

figure(1);

imshow(I);

J = dct2(I);

figure(2);

imshow(log(abs(J)),[]), colormap(jet(64)), colorbar

% 读取图像

image = imread('demo.jpg');

% 显示原图像

subplot(2, 2, 1);

imshow(image);

title('原图');

% 水平翻转

%图片的数组为hwc

% ：，end：-1：1，：

%表示宽方向中从尾开始从最后一列到第一列

horizontal\_flip = image(:, end:-1:1, :);

subplot(2, 2, 2);

imshow(horizontal\_flip);

title('水平翻转');

% 垂直翻转

vertical\_flip = image(end:-1:1, :, :);

subplot(2, 2, 3);

imshow(vertical\_flip);

title('垂直翻转');

% 中心翻转（水平+垂直）

center\_flip = image(end:-1:1, end:-1:1, :);

subplot(2, 2, 4);

imshow(center\_flip);

title('中心翻转');

sgtitle('图像的水平垂直和中心对称');

C++实现

#include <opencv2/opencv.hpp>       
#include <iostream>    
using namespace cv;  
using namespace std;  
  
int main()  
{  
    Mat src = imread("D:\\test\\5.jpg", 0);  
    imshow("原图", src);  
    src = Mat\_<float>(src);  
    int w = src.rows;  
    int h = src.cols;  
  
    int big;  
    big = 8;  
    src.convertTo(src, CV\_32F, 1.0 / 255);//转换类型  
    Mat srcDCT = Mat\_<float>(src);  
  
    for (int i1 = 0; i1 < (h / big); i1++) {  
        for (int j1 = 0; j1 < (w / big); j1++) {  
            dct(src(Rect(i1 \* big, j1 \* big, big, big)), srcDCT(Rect(i1 \* big, j1 \* big, big, big)));  
  
        }  
    }  
  
    int mask[8][8] = {  
        {1, 1, 0, 0, 0, 0, 0, 0},  
        {1, 0, 0, 0, 0, 0, 0, 0},  
        {0, 0, 0, 0, 0, 0, 0, 0},  
        {0, 0, 0, 0, 0, 0, 0, 0},  
        {0, 0, 0, 0, 0, 0, 0, 0},  
        {0, 0, 0, 0, 0, 0, 0, 0},  
        {0, 0, 0, 0, 0, 0, 0, 0},  
        {0, 0, 0, 0, 0, 0, 0, 0},  
  
    };  
    double s;  
    for (int i1 = 0; i1 < (srcDCT.rows / big); i1++)   
    {  
        for (int j1 = 0; j1 < (srcDCT.cols / big); j1++)   
        {  
            for (int i = 0; i < big; i++)  
            {  
                for (int j = 0; j < big; j++)  
                {  
                    if (mask[i % big][j % big] == 0) {  
                        srcDCT.at<int>(i1 \* big + i, j1 \* big + j) = 0;;  
                    }  
                    else {  
                        s = 0;  
                    }  
                }  
            }  
        }  
    }  
    Mat iDct1 = srcDCT;  
    for (int i1 = 0; i1 < (h / big); i1++) { //逆变换  
        for (int j1 = 0; j1 < (w / big); j1++) {  
            idct(srcDCT(Rect(i1 \* big, j1 \* big, big, big)), iDct1(Rect(i1 \* big, j1 \* big, big, big)));  
  
        }  
    }  
   
      
  
     
    imshow("进行压缩后图", srcDCT);  
    waitKey(0);  
    return 0;  
  
}