第一次作业

人工智能 2204 班 王誉诺 U202214969

1. 阐述傅里叶变换与傅里叶系数的物理含义：

傅里叶变换可以理解为将一个时间域或空间域的信号转换为频域的信号，这意味着我们可以从复杂的信号中提取出其不同的频率成分。通过傅里叶变换，任何复杂的周期信号都可以表示为若干正弦波的叠加，这些正弦波各自具有不同的频率、振幅和相位。傅里叶变换不仅揭示了信号的频率特性，还展示了信号的能量如何分布在不同的频率上，这在滤波、信号压缩等领域有很大用处。

傅里叶系数则反映了这些频率成分在信号中的权重。每个傅里叶系数的大小代表了该频率正弦波在信号中的振幅，而傅里叶系数的相位则决定了该频率正弦波在信号中的相位偏移。通过傅里叶系数，我们可以对信号进行重构，也就是说，知道了信号的傅里叶系数，我们就可以精确地还原出原始的时间域信号。傅里叶系数的大小和分布揭示了信号的频谱特征，帮助我们了解信号中的主要频率分量。

2.

（1）将图像顺时针旋转 30 度

图像旋转的本质利用的是向量的旋转，矩阵乘法的实质是进行线性变换，因此对一个向量进行旋转操作也可以通过矩阵和向量相乘的方式进行。

起初我实现代码的思路是把原图进行向量的旋转，找到旋转后的向量的位置，然后将原图的像素值赋值过去即可。但得到图像的大小和原图一样导致边缘被裁剪了，而且图像中会出现很多噪声很多杂点，出现杂点的原因是从原图旋转后的像素位置在原图可能找不到。

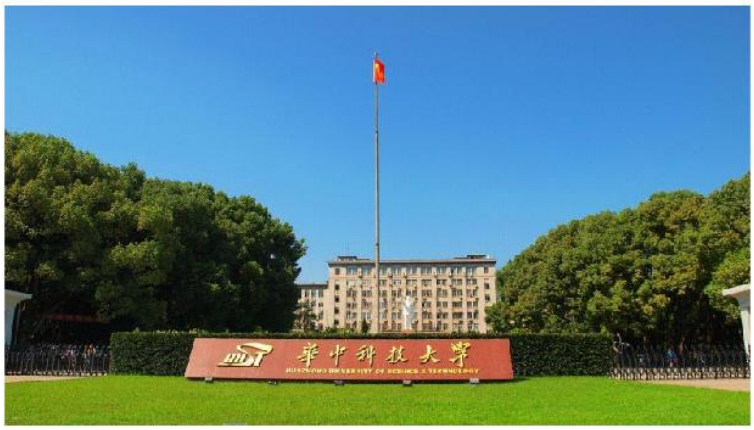
解决方法：用逆向思维，从目标图片反向旋转到原图进行像素查找。先计算出旋转后要显示完整图像所需的画布大小，然后像素赋值的顺序反过来，从目标图片反向旋转到原图进行像素查找。此外还要注意，在改变目标画布大小后，图像中心点即旋转轴改变了，要单独进行计算。

实现代码：

```
I = imread('D:\文档\数字图像处理\实验图像.bmp');
figure,imshow(I);
title('原图')

I_gray = rgb2gray(I);
figure,imshow(I_gray);
title('灰度图像')

im = I_gray;
a = 30 / 180 * pi;
R = [cos(a), sin(a); -sin(a), cos(a)];
R = R';
[h, w] = size(im);
c1 = [h; w] / 2;
hh = floor(w*sin(a) + h*cos(a)) + 1;
ww = floor(w*cos(a) + h*sin(a)) + 1;
c2 = [hh; ww] / 2;

im2 = uint8(ones(hh, ww) * 128);
for i = 1:hh
    for j = 1:ww
        p = [i; j];
        pp = R * (p - c2) + c1;
        m = floor(pp(1));
        n = floor(pp(2));
        a = pp(1) - m;
        b = pp(2) - n;
        % 检查是否在原图范围内
        if m >= 1 && m < h && n >= 1 && n < w
            % 双线性插值
            im2(i, j) = (1 - a) * (1 - b) * im(m, n) + ...
                        a * (1 - b) * im(m + 1, n) + ...
                        (1 - a) * b * im(m, n + 1) + ...
                        a * b * im(m + 1, n + 1);
        end
    end
end
figure;
imshow(im2);
title('顺时针旋转 30 度图像');
```

输出图像：

原图



灰度图像

顺时针旋转30度图像

（2）基于最近邻和双线性插值将图像分别放大 2 倍和 4 倍

a. 基于最近邻将图像放大 2 倍

最近邻插值，是将距离目标点最近的像素点的值作为插值的值。

将一张 M*N 的图像 src 放大到 P*Q 的图像 dst，我们要创建一个新的 P*Q 的数组，像素点中随便取一个点 A(i, j)，假设我们使用的灰度图，那么 A 点的灰度值:

首先把 P 点的坐标映射到原图像中，假设是 B 点，那么类比相似三角形的相似性质计算出 B 点的坐标(x, y)：(x, y) = (M/P*i, N/Q*j),由于数组的索引只能是整数，所以需要将 x 和 y 化为整数，使用最近邻插值时，可以四舍五入：

x = round(M/P*i);

y = round(N/Q*j));

dst(i, j) = src(x, y);

于是就得到了计算新图像每个像素点灰度值的方法，来一个循环就可以构建新图像。

实现代码：

```
I = imread('D:\文档\数字图像处理\实验图像.bmp');
figure,imshow(I);
title('原图')

I_gray = rgb2gray(I);
figure,imshow(I_gray);
title('灰度图像')

ratio = 2;
[row, col, color] = size(I_gray);
row = round(ratio * row);
col = round(ratio * col);

I_twice = zeros(row, col, color, class(I_gray));

for i = 1 : row
    for j = 1 : col
        x = round(i / ratio);
        y = round(j / ratio);
        if x == 0
```

```
            x = x + 1;
        end
        if y == 0
            y = y + 1;
        end
        I_twice(i, j, : ) = I_gray(x, y, :);
    end
end

figure,
imshow(I_twice);
title('放大 2 倍图像')
```

输出图像：（因插入图片篇幅限制，已将原图、灰度图及放大后所得图像等比例缩小）

原图

灰度图像

放大2倍图像

b.基于双线性插值将图像放大 4 倍

双线性插值不是只考虑距离最近的像素点，而是考虑周围四个像素点的加权值。

假设 ABCD 均为相邻的像素点，而(x, y)落在中间，插值时考虑距离的影响，离像素点越远，则权值越小，比如说 A 点，考虑到 ABCD 为边长为 1 的正方形，可以用(1-dx)(1-dy)表示 A 点灰度值的权值，其中 dx 和 dy 分别表示 x 和 y 方向的距离

假设 ABCD 处的灰度分别是 a, b, c 和 d，于是可以得到(x, y)处的灰度应该是：

$$intensity=(1-dx)(1-dy)a+(1-dx)dyb+dx(1-dy)c+dxdyd$$

在 matlab 中，ABCD 的坐标可以取整函数来获得，且在 matlab 中，图像的数组从 1 开始，应该用向上取整，并且考虑数组越界

设 A(x1, y1), B(x1, y2), C(x2, y1), D(x2, y2)

```
x1 = ceil(M/P*i);
y1 = ceil(N/Q*j);
x2 = ceil(M/P*i) + 1;
y2 = ceil(N/Q*j) + 1;
```

实现代码：

```
I = imread('D:\文档\数字图像处理\实验图像.bmp');
figure,imshow(I);
title('原图')

I_gray = rgb2gray(I);
figure,imshow(I_gray);
title('灰度图像')

ratio = 4;
[row, col, color] = size(I_gray);
row = round(ratio * row);
col = round(ratio * col);

I_four = zeros(row, col, color, class(I_gray));
for i = 1 : row
```

```
    for j = 1 : col
        x = i / ratio;
        y = j / ratio;
        x1 = ceil(x);              % 向上取整
        y1 = ceil(y);
        x2 = ceil(x) + 1;
        y2 = ceil(y) + 1;
        if x2 >= size(I_gray, 1)              % 溢出检查
            x2 = x2 - 1;
        end
        if y2 >= size(I_gray, 2)
            y2 = x2 - 1;
        end
        du = (x+1) - x1;
        dv = (y+1) - y1;
        I_four(i, j, :) = (1-du)*(1-dv)*I_gray(x1, y1, :) + (1-du)*dv*I_gray(x1, y2, :) + du*(1-dv)*I_gray(x2, y1, :) + du*dv*I_gray(x2, y2, :);
    end
end

figure,
imshow(I_twice);
title('放大 4 倍图像')
```

输出图像：（因插入图片篇幅限制，已将原图、灰度图及放大后所得图像等比例缩小）

（续接后页）

放大4倍图像

3.展开傅里叶变换

二维离散傅里叶变换 DFT 可分离性的基本思想是 DFT 可分离为两次一维 DFT。因此可以用通过计算两次一维的 fft 来得到二维快速傅里叶 fft2 算法。根据快速傅里叶变换的计算要求，需要图像的行数、列数均满足 2 的 n 次方，如果不满足，在计算 fft 之前先要对图像补零以满足 2 的 n 次。

一个 M 行 N 列的二维图像 f(x,y)，先按行队列变量 y 做一次长度为 N 的一维离散傅里叶变换，再将计算结果按列向对变量 x 做一次长度为 M 傅里叶变换就可以得到该图像的傅里叶变换结果。每一行由 N 个点，对每一行的一维 N 点序列进行离散傅里叶变换得到 F(x,u)，再对得到 F(x,u)按列向对每一列做 M 点的离散傅里叶变换，就可以得到二维图像 f(x,y)的离散傅里叶变换 F(u,v)。

且傅里叶变换对有如下平移性质：

$$f(x)e^{j2\pi xu_0/N} = F(u-u_0)$$

$f(x,y)\exp[j2\pi(u_0x/M+v_0y/N)] \Longleftrightarrow F(u-u_0,v-v_0)$

$f(x-x_0,y-y_0) \Longleftrightarrow F(u,v)\exp[-j2\pi(ux_0/M+vy_0/N)]$

式子表明，在频域中原点平移到(u0 ,v0)时，其对应的空间域 f(x,y)要乘上一个正的指数项：

$$\exp[j2\pi(u0x/M+v0y/N)]$$

在空域中图像原点平移到(x0,y0)时，其对应的F(u,v)要乘上一个负的指数项：

$$\exp[-j2\pi(ux0/M+vy0/N)]$$

在数字图像处理中，常常需要将 F(u,v)的原点移到 N*N 频域的中心，以便能清楚地分析傅里叶谱的情况，平移前空域、频域原点均在左上方。要做到这点，只需令上面平移公式中的：u0=v0=N/2；

上式表明：如果需要将图像傅里叶谱的原点从左上角(0,0)移到中心点(N/2,N/2)，只要 f(x,y)乘上$(-1)^{x+y}$因子进行傅里叶变换即可实现。

反之，当频域中 F(u,v)产生移动时，相应 f(x,y)在空域中也只发生相移，不产生幅值变化。根据平移性质，为了更清楚查看二维图像的频谱，使直流成分出项在图像中央，在把画面分成四分的基础上，进行如图所示的换位(移位)也是可以的，这样，频域原点就回平移到中心。

实现代码：

hust_fourier.m :

```
I = imread('D:\文档\数字图像处理\实验图像.bmp');
figure,imshow(I);
title('原图')

I_gray = rgb2gray(I);
figure,imshow(I_gray);
title('灰度图像')

img=double(I_gray);
F1=myfft2(img);                %傅里叶变换
```

myfft2.m :

```
function [afft2Abs,afft2Log] = myfft2(aIn)
%快速二维傅里叶变换
aDouble = im2double(aIn);
[ra,ca] = size(aDouble);
```

```
maxL = max(ra,ca);
n = 1;
while(maxL>2^n)
    n = n+1;
end
aZeros = zeros(2^n,2^n);
aZeros(1:ra,1:ca) = aDouble;
afft2 = aZeros;
[Ra,Ca] = size(afft2);
[X,Y] = meshgrid(0:Ra-1,0:Ca-1);
afft2 = afft2.*(-1).^(X+Y);
for f1 = 1:Ra
    aLine = afft2(f1,:);
    afft2(f1,:) = myfft1(aLine);
end
for f2 = 1:Ca
    aCol = afft2(:,f2);
    afft2(:,f2) = myfft1(aCol);
end
afft2Abs = abs(afft2);
afft2Log = log(afft2Abs+1);

figure,imshow(afft2Log,[]),title('傅里叶变换图（自写）');
end
```

myfft1 :

```
function fOut = myfft1(a,M)
%a 是输入的一串一维向量，M 是 2^n 大小的数值，M 可以不输入，则默认补充最邻进的那个 2^n 整数值
%快速傅里叶变换（碟式变换作为原理）
aL = length(a);
if nargin<2
    n = 1;
    while(aL > 2^n)
        n   = n+1;
    end
    aADD = zeros(1,2^n);
else
    aADD = zeros(1,M);
    n = log2(M);
end
aADD(1:aL) = a;
```

```
aL1 = length(aADD);
%对位置进行二进制取反后转为 10 进制
x = bin2dec(fliplr(dec2bin(0:aL1-1,n)))+1;
a1 = aADD(x);

%碟式变换的级数
for f1 = 1:n
    space = 2^(f1-1);
    for f2 = 0:space-1

        factor = 2^(n-f1)*f2;
        for f3 = f2+1:2^f1:aL1
            W = exp(-1j*2*pi*factor/aL1);
            temp = a1(f3)+a1(f3+space)*W;
            a1(f3+space) = a1(f3)-a1(f3+space)*W;
            a1(f3) = temp;
        end
    end
end
fOut = a1;
end
```

输出图像：

原图

灰度图像

傅里叶变换图（自写）