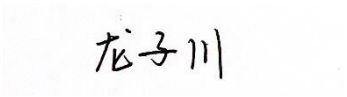
第一次作业



龙子川 人工智能 2204 班 U20221514

傅里叶变换的物理含义

傅里叶变换是一种数学工具，可用于将一个时域信号转换为频域信号。它的物理含义可以理解为对信号的频率成分进行分析。傅里叶变换提供了以下几方面的信息：

1. 频率成分：傅里叶变换可以揭示信号中存在的不同频率成分。通过分析频率成分，能够了解信号是由哪些频率的正弦波组成的。

2. 信号的能量分布：通过傅里叶变换，可以观察信号的能量在不同频率上的分布。例如，在电信号处理中，可以使用傅里叶变换来分析信号的频谱，了解哪些频率成分在信号中占主导地位。

3. 信号处理：许多信号处理技术，例如滤波、信号复原等，都是基于傅里叶变换的原理。通过在频域中处理信号，可以更有效地实现滤波或降噪。

傅里叶系数的物理含义

傅里叶系数是傅里叶级数展开中每个正弦或余弦分量的权重。

1. 幅值和相位信息：傅里叶系数不仅包含关于频率的幅值信息（表示各个频率成分在信号中所占的能量），还包含相位信息（表示各个频率成分的相位位置）。这两者结合起来可以重构原始信号。

2. 信号的周期性特征：傅里叶系数的计算是基于对周期信号的积分，其中每个系数代表了信号在特定频率上的贡献。

实验报告

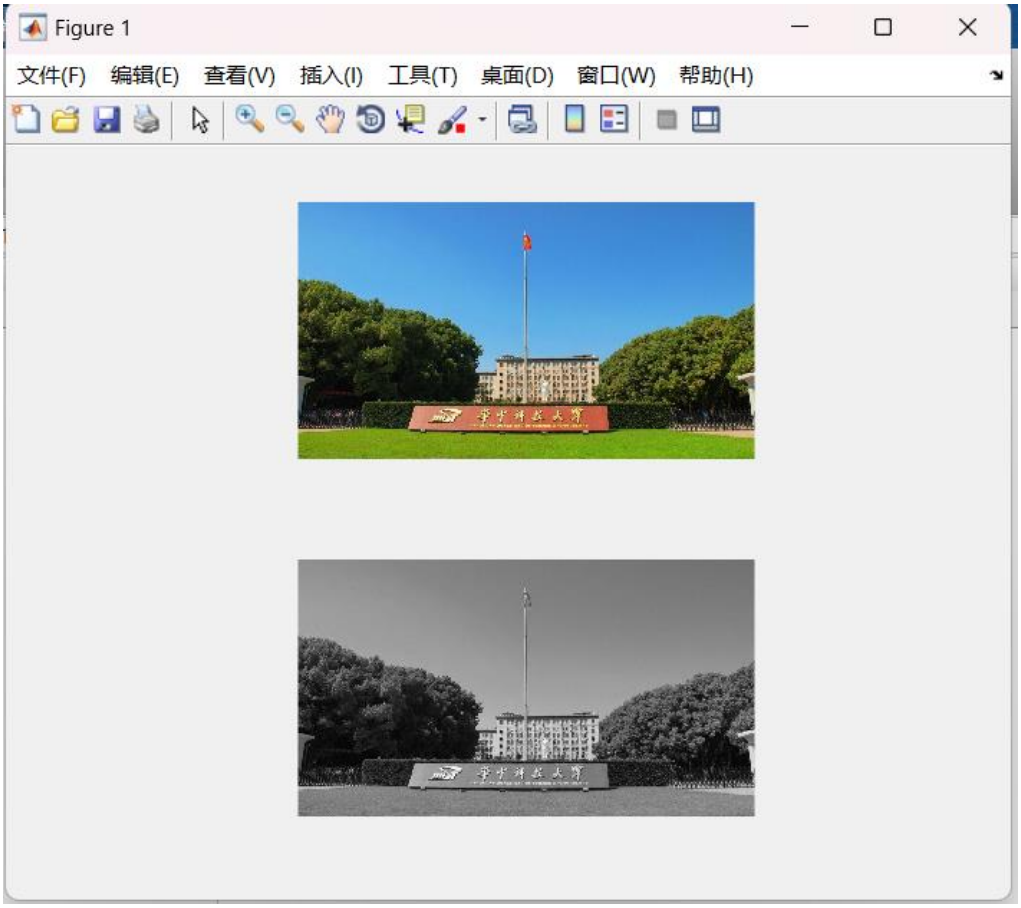
一、 灰度图转换

- **代码**

```
function picture_imread_imshow
I=imread('test_1.jpg');
% 显示原图
figure(1);
subplot(2,1,1);
imshow(I);
% 将RGB图像转换为灰度图
I_1=rgb2gray(I);
subplot(2,1,2);
```

```
imshow(I_1);
imwrite(I_1,'gray_image.png')
```

- **实验结果**



二、 顺时针旋转 30 度

- **代码**

- **最近邻**

```
function rotation(I,angle)
I=imread('gray_image.jpg');
angle=30;
[h,w,d]=size(I);
theta=angle/180*pi;
cos_val = cos(theta);
sin_val = sin(theta);

w2=round(abs(cos_val)*w+h*abs(sin_val));
h2=round(abs(cos_val)*h+w*abs(sin_val));
img_rotate  = uint8(zeros(h2,w2,3));    %无符号8位整数表示灰度图像的像素值

for x=1:w2
    for y=1:h2
```

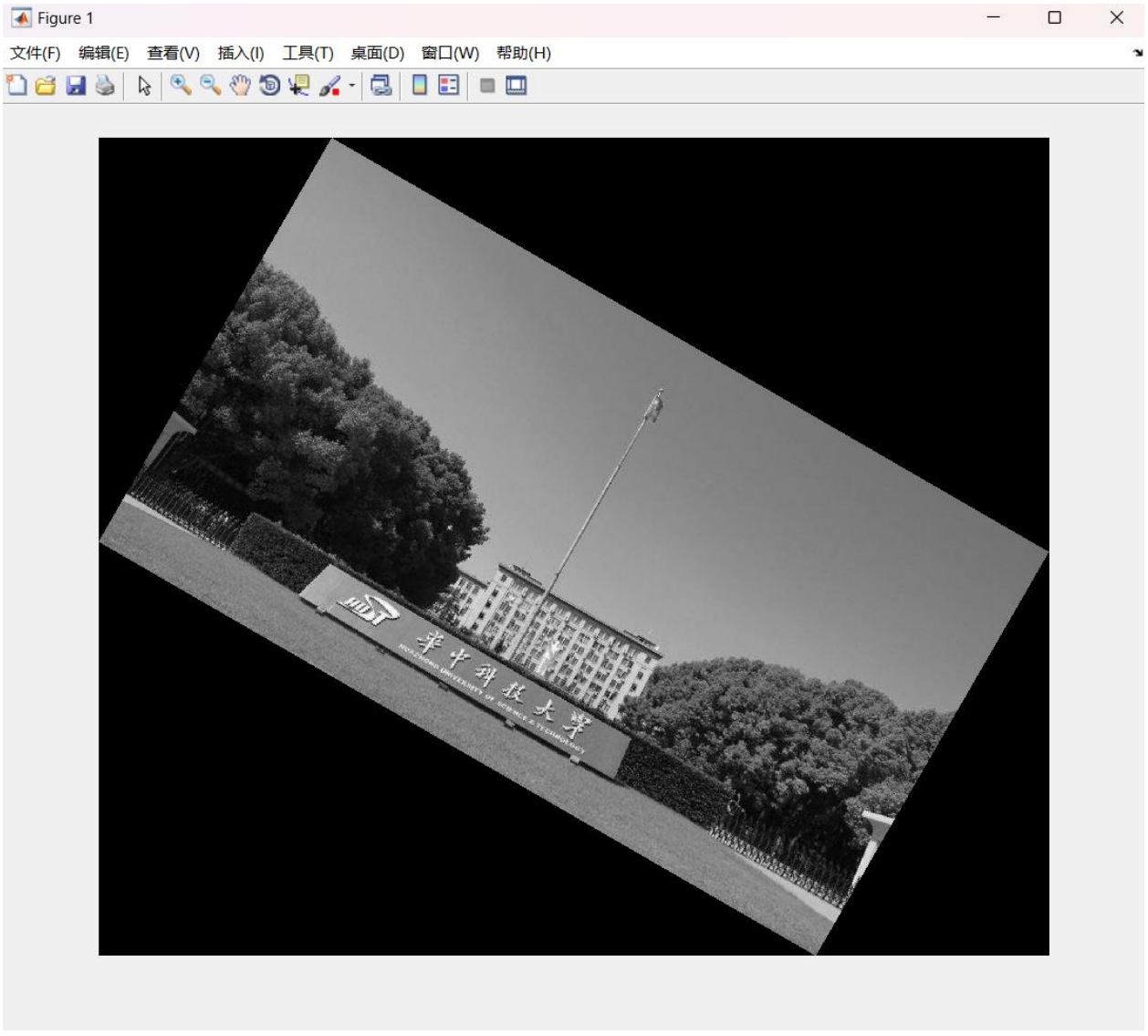
```
        %展开矩阵乘法
        x0 = uint32(x*cos_val + y*sin_val -0.5*w2*cos_val-
0.5*h2*sin_val+0.5*w);¨
        y0 = uint32(y*cos_val - x*sin_val +0.5*w2*sin_val-
0.5*h2*cos_val+0.5*h);¨
        %最近邻
        x0=round(x0);
        y0=round(y0);
        if x0>=1 && y0>=1&&  x0<=w && y0<=h %检测下标不能越界
            img_rotate(y,x,:) = I(y0,x0,:);
        end
    end
end
imshow(img_rotate);
imwrite(img_rotate,'rotation_1.jpg')
```

- 双线性插值

```
angle=30;
I=imread('gray_image.jpg');
imshow(I);
[h,w,d]=size(I);
theta=angle/180*pi;
M=[cos(theta) -sin(theta) 0;sin(theta) cos(theta) 0;0 0 1];
leftup=[1 1 1]*M;                %变换后图像左上点的坐标
rightup=[1 w 1]*M;               %变换后图像右上点的坐标
leftdown=[h 1 1]*M;              %变换后图像左下点的坐标
rightdown=[h w 1]*M;             %变换后图像右下点的坐标
cos_val = cos(theta);
sin_val = sin(theta);

w2=round(abs(cos_val)*w+h*abs(sin_val)); %变换后图像的宽度
h2=round(abs(cos_val)*h+w*abs(sin_val)); %变换后图像的高度
img_rotate=uint8(zeros(h2,w2,3));
dy=abs(min([leftup(1) rightup(1) leftdown(1) rightdown(1)]));   %取得y方向的
负轴超出的偏移量
dx=abs(min([leftup(2) rightup(2) leftdown(2) rightdown(2)]));   %取得x方向的
负轴超出的偏移量

for i=1-dy:h2-dy
    for j=1-dx:w2-dx
       pix=[I j 1]/M; %用变换后图像的点去寻找原图像的点坐标
        yy=pix(1)-floor(pix(1));
        xx=pix(2)-floor(pix(2));
        if pix(1)>=1 && pix(2)>=1 && pix(1) <= h && pix(2) <= w
            x11=floor(pix(1));       %四个相邻点
            y11=floor(pix(2));
            x12=floor(pix(1));
            y12=ceil(pix(2));
```

```
            x21=ceil(pix(1));
            y21=floor(pix(2));
            x22=ceil(pix(1));
            y22=ceil(pix(2));

            value_leftup=(1-xx)*(1-yy); %计算临近四个点在双线性插值中的权值
            value_rightup=xx*(1-yy);
            value_leftdown=(1-xx)*yy;
            value_rightdown=xx*yy;

img_rotate(i+dy,j+dx,:)=value_leftup*I(x11,y11,:)+value_rightup*I(x12,y12,
:)+ value_leftdown*I(x21,y21,:)+value_rightdown*I(x22,y22,:);
        end
    end
end
imshow(uint8(img_rotate))
```

- **实验结果**



三、 基于最近邻放大 2 倍

- **代码**

```
I = imread('gray_image.jpg');
[h, w, ch] = size(I);
new_h = round(h * 2);
new_w = round(w * 2);
large_img = zeros(new_h, new_w, ch);

for i = 1:new_h
    for j = 1:new_w
        % 计算原图像的索引
        i_1 = floor((i - 1) / 2) + 1;
        j_1 = floor((j - 1) / 2) + 1;

        % 确保索引在有效范围内
        if (i_1 >= 1 && i_1 <= h && j_1 >= 1 && j_1 <= w)
            large_img(i, j, :) = I(i_1, j_1, :); % 复制所有通道
        end
    end
end
imshow(uint8(large_img));
```

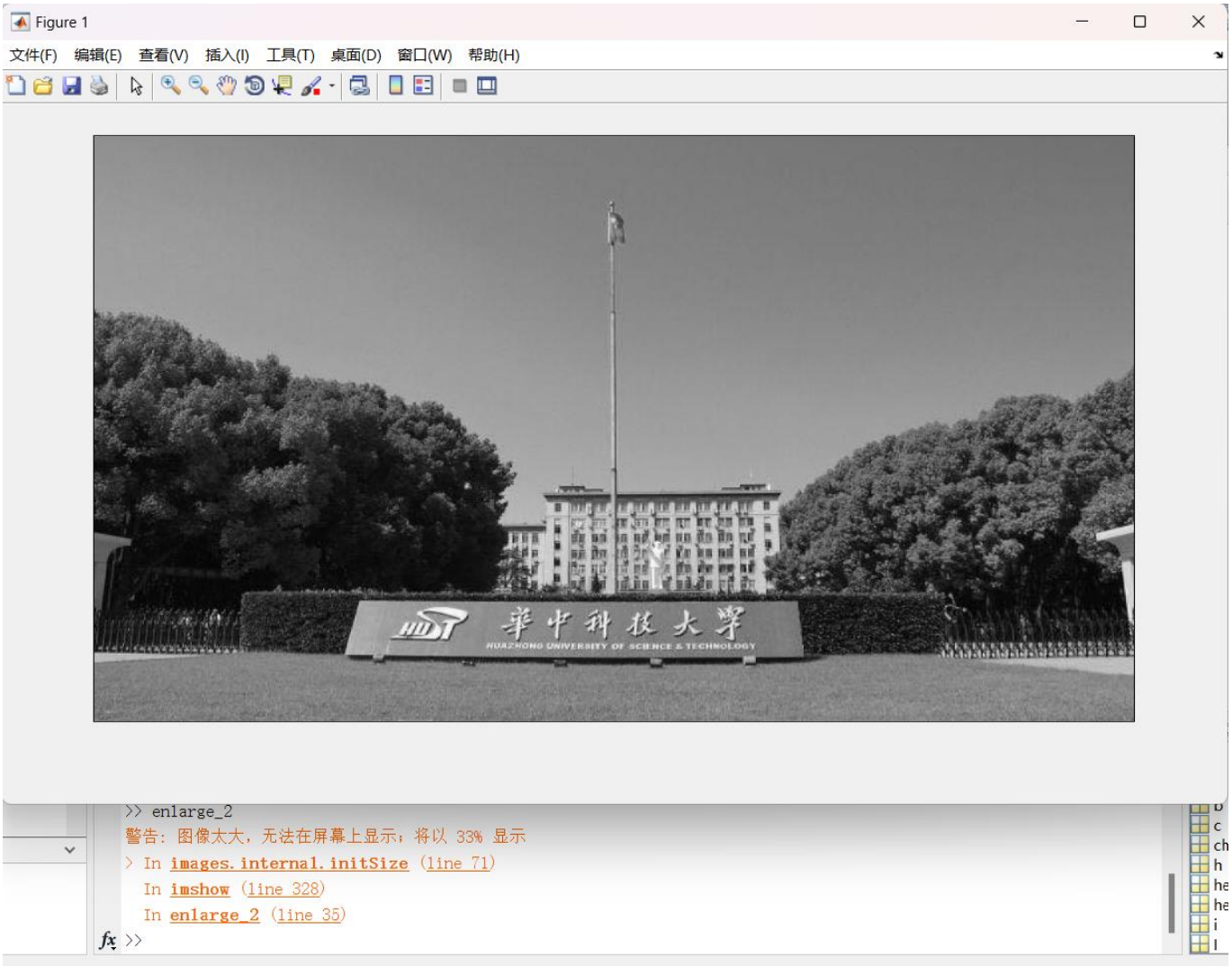
- **实验结果**

四、 基于双线性插值放大 4 倍

- **代码**

```
I = imread('gray_image.jpg');
[m, n, ch] = size(I);
new_w = round(m * 4);
new_h = round(n * 4);
large_img = uint8(zeros(new_w, new_h, ch));

for c = 1:ch
    for x = 1:new_w
        for y = 1:new_h
            % 新图像位置对应原图像位置
            xx = x / 4;
            yy = y / 4;

            % 向下取整
            a = floor(xx);
            b = floor(yy);

            if (a < 1 || b < 1 || a >= m || b >= n)  %下标不能越界
                continue;
            end

            % 处理插值
            x11 = double(I(a, b, c));  % x11 = I(a,b)
            x12 = double(I(a, min(b + 1, n), c));
            x21 = double(I(min(a + 1, m), b, c));
            x22 = double(I(min(a + 1, m), min(b + 1, n), c));

            % 双线性插值
            large_img(x, y, c) = uint8( (b + 1 - yy) * ((xx - a) * x21 + (a + 1
- xx) * x11) + ...
                                        (yy - b) * ((xx - a) * x22 + (a + 1 - xx)
* x12));
        end
    end
end

imshow(large_img);
```

- **实验结果**



五、 傅里叶变换和平移

- **代码**

```
function imageDFT()
    I=imread('gray_image.jpg');
    I=im2double(I);
    [x,y] = size(I);
    Ax = ones(x,y);
    A = ones(x,y);
    com = 0+1i;
    % 对每一列进行DFT
    for k =1:x
        for m=1:y
            sn =0;
            for n =1:x
                sn =sn + I(n,m)*exp(-com*2*pi*k*n/x);
            end
            Ax(k,m) = sn;
        end
```

```
    end
    % 对每一行进行DFT
    for p =1:y
        for k =1:x
            sn =0;
            for m=1:y
                sn = sn+Ax(k,m)*exp(-com*2*pi*p*m/y);
            end
            A(k,p) = sn;
        end
    end
    F=my_fftshift(A);
    F= abs(F);
    F=log(F+1);
    imshow(F,[]);
end

function y = my_fftshift(x)

    sz = size(x);
    % 对于二维数组
    if length(sz) == 2
        % 对行列都进行 fftshift
        y = fftshift2d(x, sz(1), sz(2));
    % 对于一维数组
    else
        N = length(x);
        if mod(N, 2) == 0
            % 将前半部分和后半部分进行置换
            y = [x(N/2 + 1:end); x(1:N/2)];
        else
            % N为奇数的情况
            y = [x(ceil(N/2) + 1:end); x(1:floor(N/2)); x(ceil(N/2))];
        end
    end
end

function y = fftshift2d(x, M, N)
    % 二维数组进行行和列的 fftshift

    y = x;
    % 行处理
    if mod(M, 2) == 0
        % 偶数行
        y = [y(M/2 + 1:end, :); y(1:M/2, :)];
    else
        % 奇数行
        y = [y(ceil(M/2) + 1:end, :); y(1:floor(M/2), :); y(ceil(M/2), :)];
```

```
    end

    % 列处理
    if mod(N, 2) == 0
        % 偶数列
        y = [y(:, N/2 + 1:end), y(:, 1:N/2)];
    else
        % 奇数列
        y = [y(:, ceil(N/2) + 1:end), y(:, 1:floor(N/2)), y(:, ceil(N/2))];
    end
end
```

- **实验结果**