华中科技大学

数字图像处理课后作业

第一次作业

学　　号: U202215138

姓　　名: 康王梓宝

院系: 人工智能与自动化学院

班　　级: 人工智能 2204 班

报告日期: 2024.9.21

目录

一、实验目的

1. **学习并掌握将彩色图像转换为灰度图像的基本方法**
2. **利用旋转矩阵实现图像的顺时针旋转**
3. **理解并实现基于最近邻插值和双线性插值的图像放大技术**
4. **深入理解傅里叶变换与傅里叶系数的物理含义，学习如何将时域信号转换为频域信号**

二、实验环境

Matlab 仿真

三、实验内容

1. **阐述傅里叶变换与傅里叶系数的物理含义**

傅里叶变换将**时域信号**转换为**频域信号**，揭示了信号在各个频率上的分布情况。通过将信号分解为不同频率的正弦和余弦波，展示了信号的**频率成分**。包括:

频谱（Spectrum）:

- F(ω)描述了信号在不同频率上的能量分布，称为频谱。

频率成分（Frequency Components）:

- 每个频率 ω 对应的 F(ω)表示该频率成分在信号中的振幅和相位。

2. 旋转与放大

3. 傅里叶

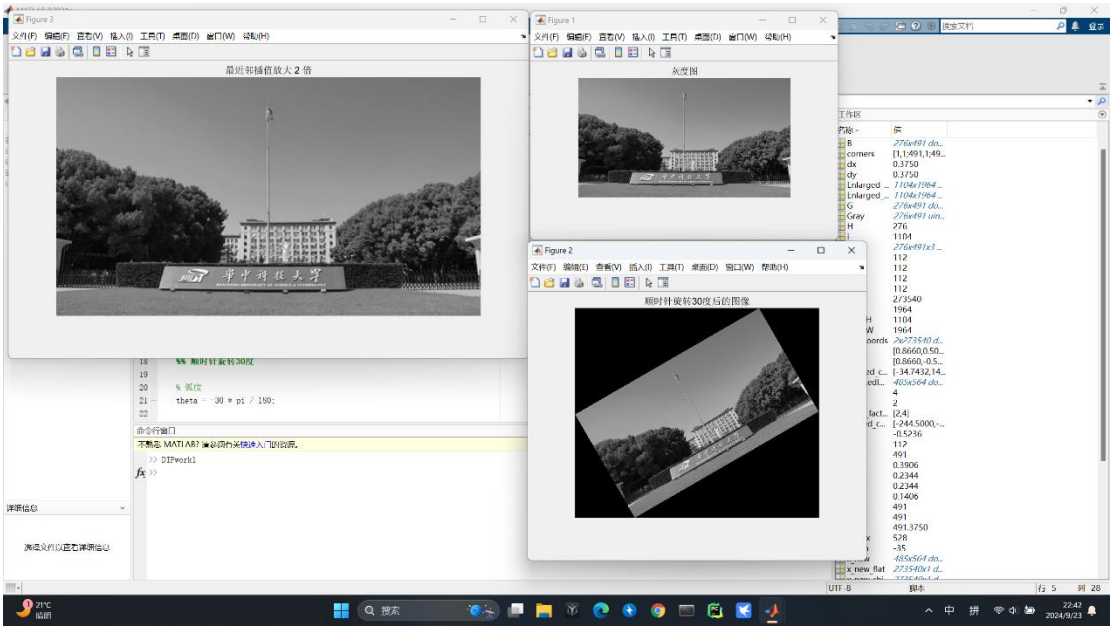


Figure 3
最近邻插值放大 2 倍
Figure 1
灰度图
Figure 2
顺时针旋转30度后的图像
华中科技大学

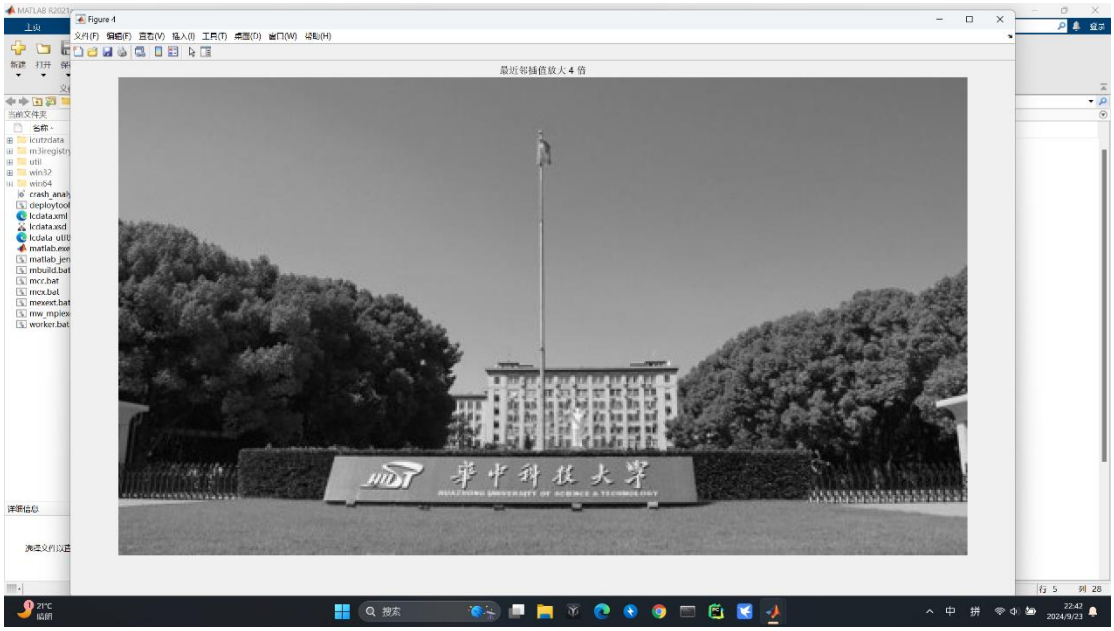


Figure 4
最近邻插值放大 4 倍
华中科技大学

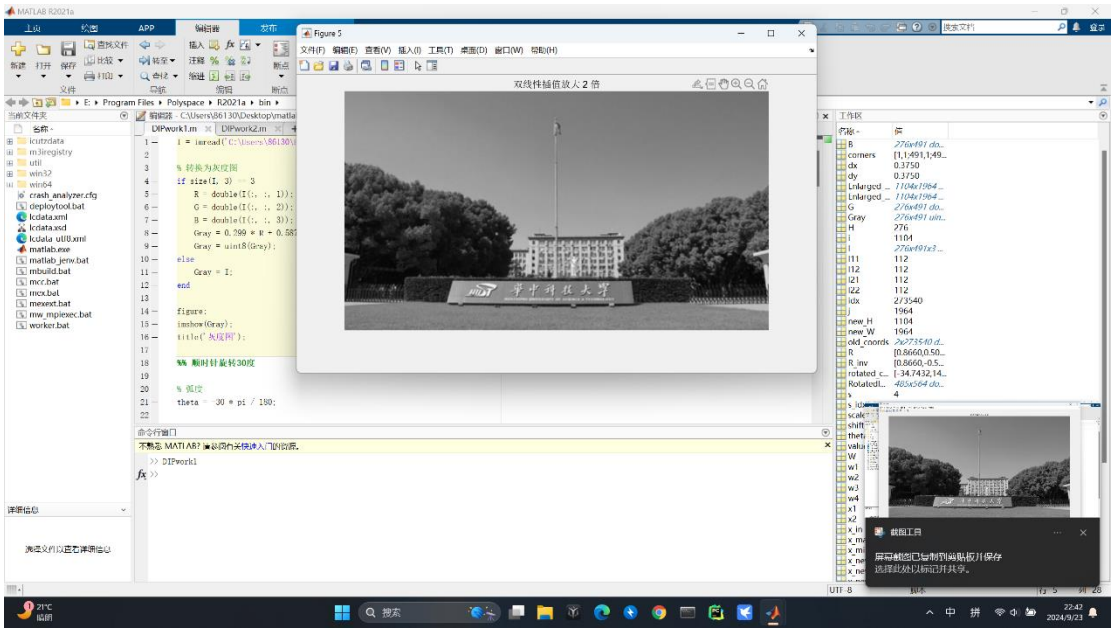


Figure 5
双线性插值放大 2 倍
截图工具
屏幕截图已复制到剪贴板并保存
选择此处以标记并共享。

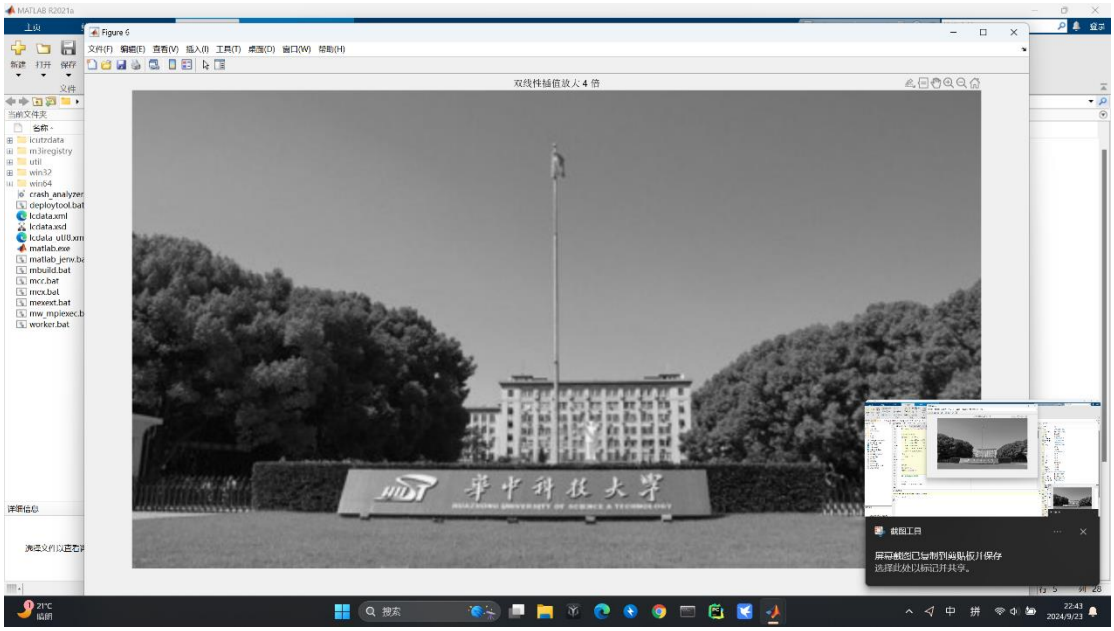
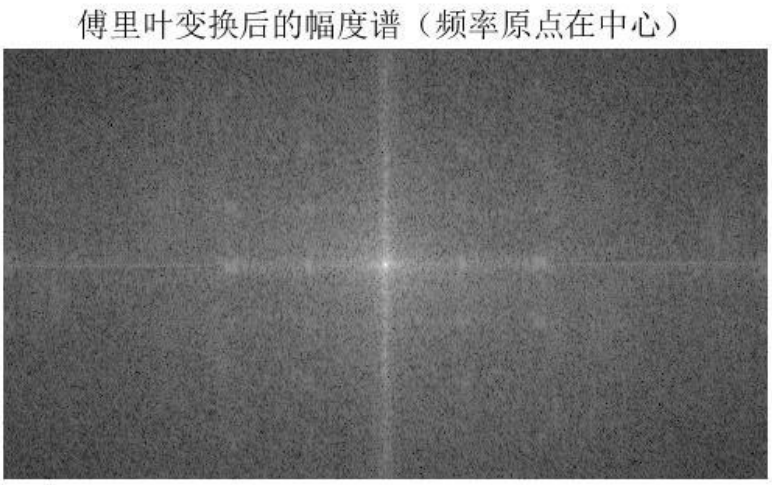


Figure 6
双线性插值放大 4 倍
截图工具
屏幕截图已复制到剪贴板并保存
选择此处以标记并共享。

4. 二维傅里叶变换



四、源代码

1. 旋转与插值

```
I = imread('C:\Users\86130\Pictures\1.jpg');

% 转换为灰度图
if size(I, 3) == 3
    R = double(I(:, :, 1));
    G = double(I(:, :, 2));
    B = double(I(:, :, 3));
    Gray = 0.299 * R + 0.587 * G + 0.114 * B;
    Gray = uint8(Gray);
else
    Gray = I;
end

figure;
imshow(Gray);
title('灰度图');

%% 顺时针旋转 30 度
```

```
% 弧度
theta = -30 * pi / 180;

% 灰度图像的尺寸
[H, W] = size(Gray);

% 图像中心
xc = W / 2;
yc = H / 2;

% 计算旋转后的图像尺寸
corners = [1, 1;
           W, 1;
           W, H;
           1, H];

shifted_corners = corners - [xc, yc];
R = [cos(theta), -sin(theta);
     sin(theta), cos(theta)];
rotated_corners = (R * shifted_corners')' + [xc, yc];

x_min = floor(min(rotated_corners(:,1)));
x_max = ceil(max(rotated_corners(:,1)));
y_min = floor(min(rotated_corners(:,2)));
y_max = ceil(max(rotated_corners(:,2)));

new_W = x_max - x_min + 1;
new_H = y_max - y_min + 1;

% 创建新图像的坐标
[x_new, y_new] = meshgrid(x_min:x_max, y_min:y_max);
x_new_flat = x_new(:);
y_new_flat = y_new(:);

% 换回原图像坐标
x_new_shifted = x_new_flat - xc;
y_new_shifted = y_new_flat - yc;

R_inv = [cos(-theta), -sin(-theta);
         sin(-theta), cos(-theta)];

old_coords = R_inv * [x_new_shifted'; y_new_shifted'];
x_old_flat = old_coords(1,:)' + xc;
y_old_flat = old_coords(2,:)' + yc;
```

```
% 初始化
RotatedImage = zeros(new_H, new_W);

% 最近邻插值
for idx = 1:length(x_new_flat)
    x_old = x_old_flat(idx);
    y_old = y_old_flat(idx);
    x_in = round(x_old);
    y_in = round(y_old);
    if x_in >=1 && x_in <= W && y_in >=1 && y_in <= H
        RotatedImage(y_new_flat(idx) - y_min + 1, x_new_flat(idx) - x_min + 1) =
Gray(y_in, x_in);
    else
        RotatedImage(y_new_flat(idx) - y_min + 1, x_new_flat(idx) - x_min + 1) = 0;
    end
end

figure;
imshow(uint8(RotatedImage));
title('顺时针旋转 30 度后的图像');

%% 最近邻插值放大 2 倍和 4 倍

scale_factors = [2, 4];

for s_idx = 1:length(scale_factors)
    s = scale_factors(s_idx);
    new_H = s * H;
    new_W = s * W;
    Enlarged_NN = zeros(new_H, new_W);
    for i = 1:new_H
        for j = 1:new_W
            x_in = round((j - 0.5) / s + 0.5);
            y_in = round((i - 0.5) / s + 0.5);
            if x_in >= 1 && x_in <= W && y_in >= 1 && y_in <= H
                Enlarged_NN(i, j) = Gray(y_in, x_in);
            else
                Enlarged_NN(i, j) = 0;
            end
        end
    end

    figure;
```

```
    imshow(uint8(Enlarged_NN));
    title(['最近邻插值放大 ', num2str(s), ' 倍']);
end

%% 双线性插值放大 2 倍和 4 倍

for s_idx = 1:length(scale_factors)
    s = scale_factors(s_idx);
    new_H = s * H;
    new_W = s * W;
    Enlarged_BL = zeros(new_H, new_W);
    for i = 1:new_H
        for j = 1:new_W
            x_in = (j - 0.5) / s + 0.5;
            y_in = (i - 0.5) / s + 0.5;

            x1 = floor(x_in);
            x2 = ceil(x_in);
            y1 = floor(y_in);
            y2 = ceil(y_in);

            % 边界检查
            x1 = max(1, min(W, x1));
            x2 = max(1, min(W, x2));
            y1 = max(1, min(H, y1));
            y2 = max(1, min(H, y2));

            dx = x_in - x1;
            dy = y_in - y1;

            I11 = double(Gray(y1, x1));
            I12 = double(Gray(y1, x2));
            I21 = double(Gray(y2, x1));
            I22 = double(Gray(y2, x2));

            w1 = (1 - dx) * (1 - dy);
            w2 = dx * (1 - dy);
            w3 = (1 - dx) * dy;
            w4 = dx * dy;

            value = w1 * I11 + w2 * I12 + w3 * I21 + w4 * I22;
            Enlarged_BL(i, j) = value;
        end
    end
```

```
    figure;
    imshow(uint8(Enlarged_BL));
    title(['双线性插值放大 ', num2str(s), ' 倍']);
end
```

2. 傅里叶变换

```
% 读取图像并转换为灰度图
I = imread('C:\Users\86130\Pictures\1.jpg');

% 将图像转换为灰度图
if size(I, 3) == 3
    % 如果是 RGB 图像，手动转换为灰度
    R = double(I(:, :, 1));
    G = double(I(:, :, 2));
    B = double(I(:, :, 3));
    Gray = 0.299 * R + 0.587 * G + 0.114 * B;
    Gray = uint8(Gray);
else
    % 如果已经是灰度图，直接使用
    Gray = I;
end

% 显示灰度图像
figure;
imshow(Gray);
title('灰度图像');

% 对灰度图像进行傅里叶变换
F = fft2(double(Gray));

% 将频率原点移至图像中心
F_shifted = fftshift(F);

% 计算傅里叶变换的幅度谱
F_magnitude = abs(F_shifted);

% 对幅度谱取对数，增强显示效果
F_log = log(1 + F_magnitude);

% 显示傅里叶变换后的幅度谱
figure;
imshow(mat2gray(F_log));
title('傅里叶变换后的幅度谱（频率原点在中心）');
```