



华中科技大学
人工智能与自动化学院
SCHOOL OF ARTIFICIAL INTELLIGENCE AND AUTOMATION, HUST

数字图像处理

第一次作业报告

班级： 人工智能 2204 班

姓名： 黄佳颖

学号： U202215167

一、问题一

- 题目：将所给图片导入，转为灰度图后，并顺时针旋转 30 度。

- 问题分析：

图像的旋转变换主要是将像素点坐标与旋转矩阵相乘，得到新的坐标后将新的坐标赋予相同的灰度值，以完成图像的旋转。

需要注意的是，由于与旋转矩阵相乘后得到的坐标不一定是整数，所以会对旋转后坐标进行取整处理，这也导致了旋转后的图片像素点之间存在间隙。

此外，由于设置旋转后图片与之前相同，所以在旋转过程中，有些旋转后的坐标将会超出图片坐标系，因而无法被看见。旋转后的图片中也有一些坐标由于没有旋转前的坐标相对应，因而呈现黑色。

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- 结果图片：



如图，与之前分析相同，旋转 30° 后的图片像素点中间存在间隙，且图片部分被遮挡，部分为黑色。

● 代码:

```
img = imread('1.bmp'); % 读取文件

% 旋转图片 30 度
rotatedImage = rotateImageSameSize(img, 30);

% 显示原始和旋转后的图像
figure;
subplot(1, 2, 1), imshow(rgb2gray(img)), title('原图');
subplot(1, 2, 2), imshow(rotatedImage), title('旋转 30°');

function rotatedImage = rotateImageSameSize(image, angle)
    % 转换为灰度图
    if size(image, 3) == 3
        grayImage = rgb2gray(image);
    else
        grayImage = image;
    end

    % 获取原图尺寸
    [height, width] = size(grayImage);

    % 初始化旋转后的图像
    rotatedImage = zeros(height, width, 'like', grayImage);

    % 旋转中心
    centerX = width / 2;
    centerY = height / 2;

    % 转换角度为弧度
    radianAngle = deg2rad(angle);

    % 遍历原始图像的每个像素
    for y = 1:height
        for x = 1:width
            % 计算旋转后的坐标
            newX = round((cos(radianAngle) * (x - centerX) -
sin(radianAngle) * (y - centerY)) + centerX);
            newY = round((sin(radianAngle) * (x - centerX) +
cos(radianAngle) * (y - centerY)) + centerY);

            % 检查新坐标是否在旋转后图像的边界内
            if newX >= 1 && newX <= width && newY >= 1 && newY <= height
```

```

        % 赋值
        rotatedImage(newY, newX) = grayImage(y, x);
    end
end
end
end
end

```

二、问题二

- **题目：**将所给图片导入，转为灰度图后，并基于最近邻和双线性插值将图像分别放大 2 倍和 4 倍。

- **问题分析：**

与旋转同样的，由于放大后的新坐标不能完全覆盖图片上的所有坐标点，导致像素点之间会存在间隙。为了填补间隙，这里采用两种不同的灰度插值法，如下。

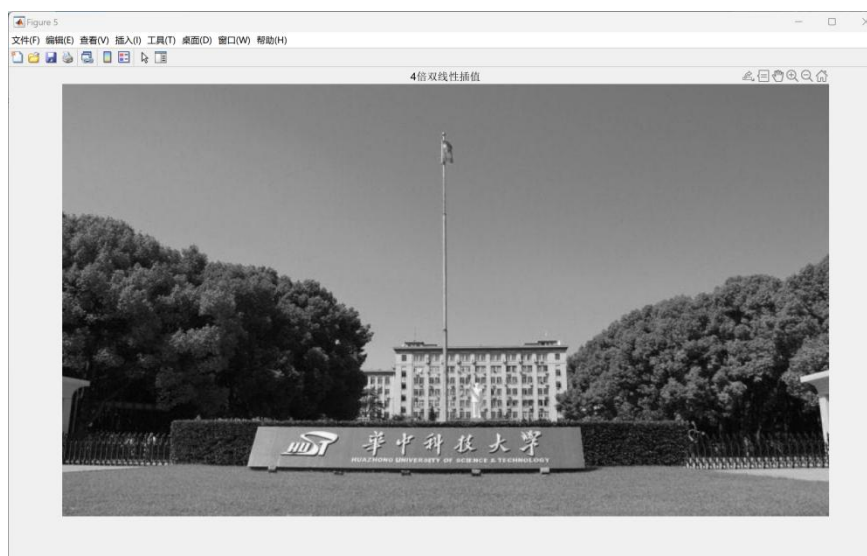
▶ **最近邻插值法：**也称作零阶插值，就是令变换后像素的灰度值等于距它最近的输入像素的灰度值。

▶ **双线性插值：**也称作一阶插值，该方法通常是沿图像矩阵的每一列（行）进行插值，然后对插值后所得到的矩阵再沿着行（列）方向进行线性插值。

- **结果图片：**



由于 subplot 函数显示将三张图放在了一个界面，且可以用鼠标滑动缩放，所以效果不是很明显。于是我又使用 imshow 单独显示了三张图，以展示放大效果和放大后采用插值产生的人工痕迹。



由于撰写报告时对截图有缩放的原因，所以 2 倍和 4 倍的图像放大关系没有这么直观，但还是可以看到图像放大的痕迹，和由于插值产生的人工痕迹。

● 代码：

```
img = imread('1.bmp'); % 读取 RGB 图像
resizedImg2x = resizeGrayImage(img, 2, 'nearest'); % 放大 2 倍，使用最近邻插值
resizedImg4x = resizeGrayImage(img, 4, 'bilinear'); % 放大 4 倍，使用双线性插值

% 显示结果
figure;
subplot(1, 3, 1); imshow(rgb2gray(img)); title('原图');
subplot(1, 3, 2); imshow(resizedImg2x); title('2 倍最近邻插值');
subplot(1, 3, 3); imshow(resizedImg4x); title('4 倍双线性插值');
function [resizedImg] = resizeGrayImage(img, scale, method)
    % 将 RGB 图像转换为灰度图像
    grayImg = rgb2gray(img);

    % 灰度图像尺寸
    [rows, cols] = size(grayImg);

    % 目标图像尺寸
    newRows = rows * scale;
    newCols = cols * scale;

    % 初始化目标图像
    resizedImg = zeros(newRows, newCols, 'uint8');

    % 放大处理
    if strcmp(method, 'nearest')
        for i = 1:newRows
            for j = 1:newCols
                % 计算最近邻点
                srcRow = round(i / scale);
                srcCol = round(j / scale);

                % 确保索引在范围内
                srcRow = min(max(srcRow, 1), rows);
                srcCol = min(max(srcCol, 1), cols);

                % 赋值
```

```

        resizedImg(i, j) = grayImg(srcRow, srcCol);
    end
end
elseif strcmp(method, 'bilinear')
    for i = 1:newRows
        for j = 1:newCols
            % 计算周围的四个点
            x = (j - 1) / scale + 1;
            y = (i - 1) / scale + 1;
            x1 = floor(x);
            y1 = floor(y);
            x2 = ceil(x);
            y2 = ceil(y);

            % 确保索引在范围内
            x1 = min(max(x1, 1), cols);
            x2 = min(max(x2, 1), cols);
            y1 = min(max(y1, 1), rows);
            y2 = min(max(y2, 1), rows);

            % 双线性插值
            u = x - x1;
            v = y - y1;
            resizedImg(i, j) = ...
                (1-u)*(1-v)*grayImg(y1, x1) + ...
                (1-u)*v*grayImg(y2, x1) + ...
                u*(1-v)*grayImg(y1, x2) + ...
                u*v*grayImg(y2, x2);
        end
    end
end
end
end

```

三、问题三

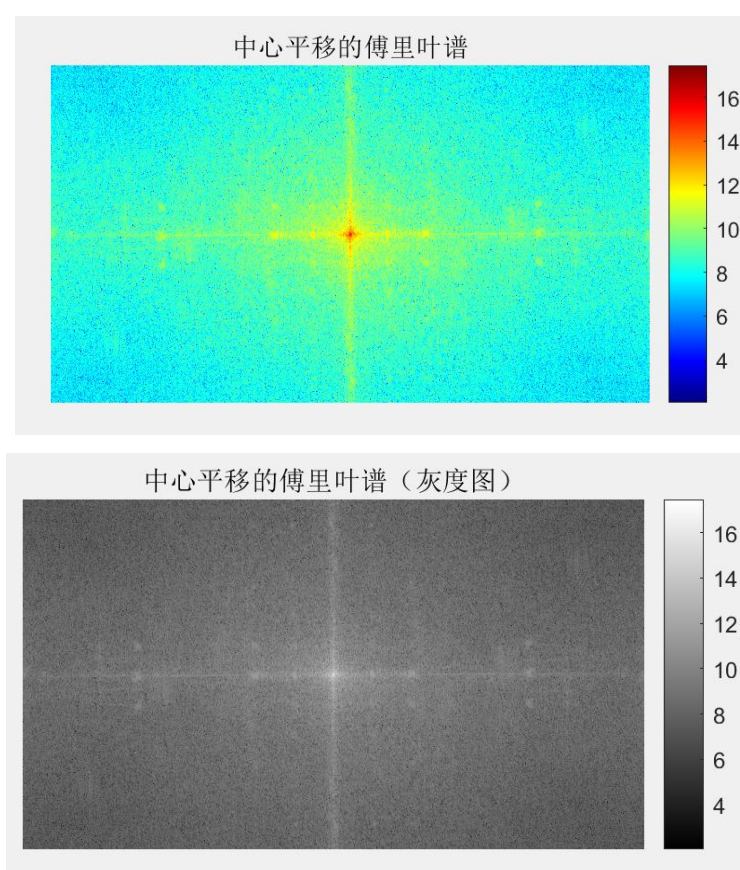
- **题目：**编写程序导入图片，自行转换为灰度图，并展开傅里叶变换，提取傅里叶变换图像（将频率原点移至图像中心）
- **问题分析：**
傅里叶变换是一种线性的积分变换，常在将信号在时域（或空域）和频域之

间变换时使用。下为对 $N \times N$ 的图像进行二维离散傅里叶变换的公式。

$$F(u, v) = \frac{1}{NN} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp\left(-j2\pi\left(\frac{ux}{N} + \frac{vy}{N}\right)\right)$$

在 MATLAB 中，我们常使用函数 `fft` 进行一维离散傅里叶变换（DFT）、函数 `fft2` 进行二维 DFT。

- 结果图片：



这里我们分别给出了 `rgb` 形式的傅里叶谱图和灰度图转成的傅里叶谱图。如图，可知达成了中心平移和傅里叶变换的任务要求。

- 代码：

```
% 读取图片  
img = imread('1.bmp');
```



```

% 转换为灰度图
grayImg = rgb2gray(img);

% 展示原始灰度图
figure;
imshow(grayImg);
title('原图');

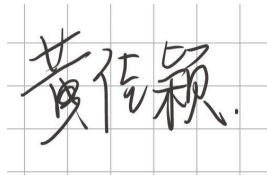
% 对灰度图进行傅里叶变换
% 使用 fftshift 来将零频率分量移到频谱中心
F = fft2(double(grayImg));
Fshift = fftshift(F);

% 计算频谱的幅度谱
magnitude_spectrum = log(1 + abs(Fshift));

% 展示频谱图像
figure;
imshow(magnitude_spectrum, []); % [] 自动调整显示范围
title('中心平移的傅里叶谱');
colormap(jet); % 使用 jet 颜色映射来更好地显示频谱
colorbar; % 显示颜色条

```

电子签名：



Handwritten signature of Huang Yanyan (黄彦颖) on a grid background.