傅里叶变换揭示了图像信号中的频率成分，即分析一个信号可以看作是多个不同频率的正弦波叠加的结果，系数代表着频率成分中每种成分所占权重。

（1）

**实验内容**：第一题编写程序（建议 Matlab）对以上图像（自行转换为灰度图）展开（1）顺时针旋转 30 度；（2）基于最近邻和双线性插值将图像分别放大 2 倍和 4 倍。

**实验思路**：按照空间变换定义相应的变换矩阵，求取旋转后的图像，采用最近邻插值来进行填补。最近邻插值放大图像，利用放大后的图像反向求解得到一个浮点数，浮点数最近邻的原图像像素点即为所求的插值。双线性插值也是通过目标图像像素点求得原图像对应位置，再求出周围四个像素点即可，注意应该判断是否为图像边缘的点，避免数组溢出。

**实验代码**：

```
img = imread('实验图像.bmp');

gray_img = rgb2gray(img);

figure()

imshow(gray_img);title('灰度图');



% 求出旋转矩阵

a = 30;

R = [cosd(a), -sind(a); sind(a), cosd(a)];

R = R'; % 求出旋转矩阵的逆矩阵进行逆向查找



% 计算原图大小

size_img = size(gray_img);

h = size_img(1);

w = size_img(2);

c1 = [h; w] / 2;



% 计算显示完整图像需要的画布大小

hh = floor(w*sind(a)+h*cosd(a))+1;

ww = floor(w*cosd(a)+h*sind(a))+1;

c2 = [hh; ww] / 2;



% 初始化目标画布
```

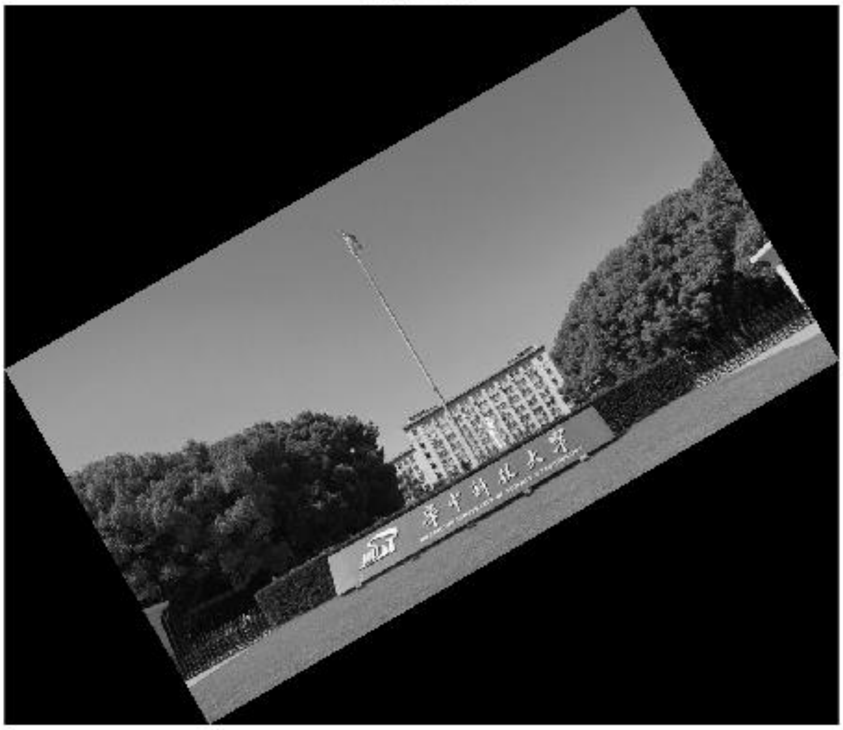
```
im2 = uint8(zeros(hh, ww));

for i = 1:hh
    for j = 1:ww
        p = [i; j];
        pp = round(R*(p-c2)+c1);
        % 逆向进行像素查找
        if (pp(1) >= 1 && pp(1) <= h && pp(2) >= 1 && pp(2) <= w)
            im2(i, j) = gray_img(pp(1), pp(2));
        end
    end
end

% 显示图像
figure()
imshow(im2);title('旋转 30 度');

% 最近邻插值放大函数
function enlarged_img = enlarge_img(original_img, scale)
    [orig_rows, orig_cols] = size(original_img);
    new_rows = round(orig_rows * scale);
    new_cols = round(orig_cols * scale);
    enlarged_img = uint8(zeros(new_rows, new_cols));

    for i = 1:new_rows
        for j = 1:new_cols
            orig_x = round(j / scale);
            orig_y = round(i / scale);
            if orig_x > 0 && orig_x <= orig_cols && orig_y > 0 && orig_y <= orig_rows
```

```
                enlarged_img(i, j) = original_img(orig_y, orig_x);

            end

        end

    end

end


% 最近邻插值放大 2 倍和 4 倍

enlarged_nn_2x = enlarge_img(gray_img, 2);

figure; imshow(enlarged_nn_2x); title('最近邻插值放大 2 倍');


enlarged_nn_4x = enlarge_img(gray_img, 4);

figure; imshow(enlarged_nn_4x); title('最近邻插值放大 4 倍');


% 双线性插值放大函数

function enlarged_img = bilinear_enlarge(original_img, scale)

    [orig_rows, orig_cols] = size(original_img);

    new_rows = round(orig_rows * scale);

    new_cols = round(orig_cols * scale);

    enlarged_img = uint8(zeros(new_rows, new_cols));


    for i = 1:new_rows

        for j = 1:new_cols

            orig_x = (j - 0.5) / scale + 0.5;

            orig_y = (i - 0.5) / scale + 0.5;

            x1 = floor(orig_x); x2 = ceil(orig_x);

            y1 = floor(orig_y); y2 = ceil(orig_y);


            if x1 > 0 && x1 <= orig_cols && y1 > 0 && y1 <= orig_rows

                q11 = double(original_img(y1, x1));
            else
                q11 = 0;
            end

            if x2 > 0 && x2 <= orig_cols && y1 > 0 && y1 <= orig_rows
                q12 = double(original_img(y1, x2));
            else
                q12 = 0;
            end

            if x1 > 0 && x1 <= orig_cols && y2 > 0 && y2 <= orig_rows
                q21 = double(original_img(y2, x1));
            else
                q21 = 0;
            end

            if x2 > 0 && x2 <= orig_cols && y2 > 0 && y2 <= orig_rows
                q22 = double(original_img(y2, x2));
            else
                q22 = 0;
            end

            % 双线性插值
            top = q11 * (x2 - orig_x) + q12 * (orig_x - x1);
            bottom = q21 * (x2 - orig_x) + q22 * (orig_x - x1);
            enlarged_img(i, j) = uint8(top * (y2 - orig_y) + bottom * (orig_y - y1));
        end
    end
end
```

```
% 基于双线性插值放大 2 倍和 4 倍

enlarged_bilinear_2x = bilinear_enlarge(gray_img, 2);

figure; imshow(enlarged_bilinear_2x); title('双线性插值放大 2 倍');



enlarged_bilinear_4x = bilinear_enlarge(gray_img, 4);

figure; imshow(enlarged_bilinear_4x); title('双线性插值放大 4 倍');
```

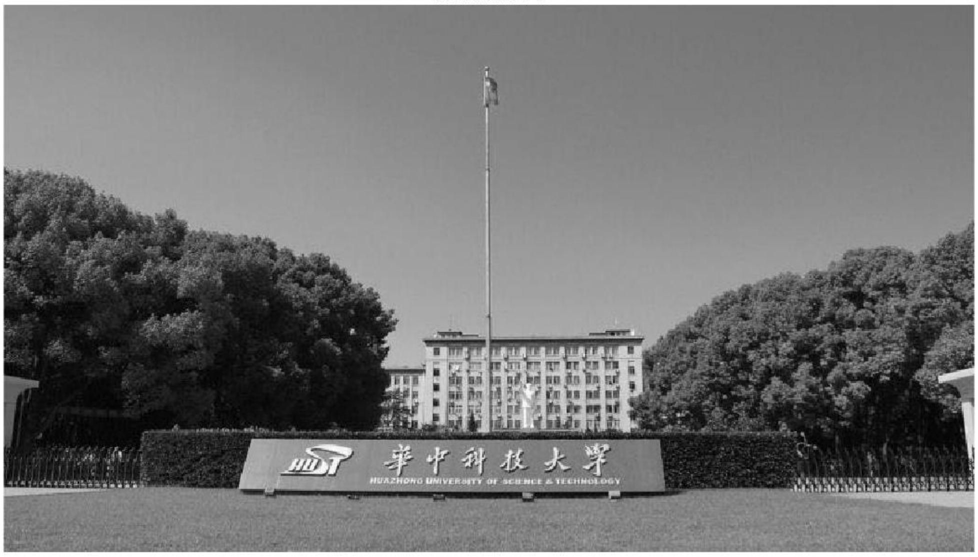
灰度图

旋转30度

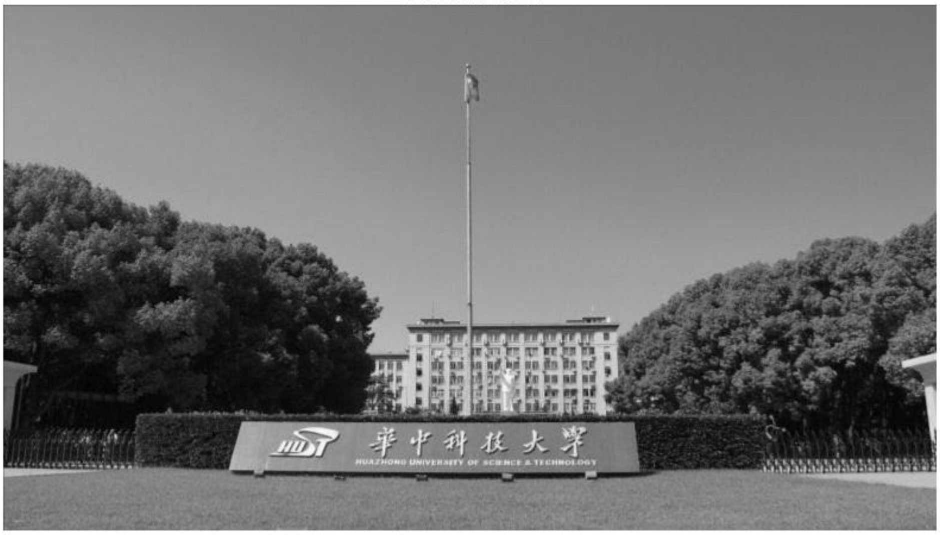


最近邻插值放大2倍

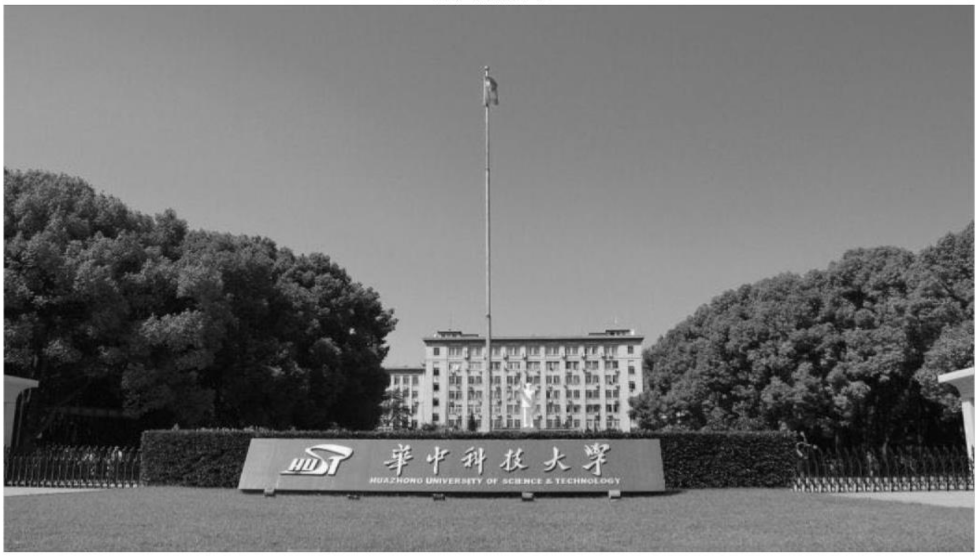
最近邻插值放大4倍



双线性插值放大2倍



双线性插值放大4倍



（2）

**实验内容**：第二题编写程序（建议 Matlab）对以上图像（自行转换为灰度图）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）

**实验思路**：直接使用 4 层嵌套计算时间过长，利用傅里叶变换可分离性拆为三层嵌套计算，频率原点利用公式

$$f(x,y)(-1)^{x+y} \Leftrightarrow F\left(u-\frac{N}{2},v-\frac{N}{2}\right)$$

进行迁移。

**实验代码**：

```
% 读取图像并转换为灰度图

    img = imread('实验图像.bmp');

    gray_img = rgb2gray(img);

    figure();

    imshow(gray_img);

    gray_img = im2double(gray_img);

    [x, y] = size(gray_img);



    gray_img=gray_img*(-1)^(x+y);



    % 对每一列进行 DFT

    AA = zeros(x, y);
```

```
for m = 1:y
    for n = 1:x
        sumCol = 0;
        for u = 1:x
            sumCol = sumCol + gray_img(u, m) * exp(-1i * 2 * pi * ((n - 1) * (u - 1)/x));
        end
        AA(n, m) = sumCol;
    end
end

% 对每一行进行 DFT
BB = zeros(x, y);
for k = 1:x
    for l = 1:y
        sumRow = 0;
        for v = 1:y
            sumRow = sumRow + AA(k, v) * exp(-1i * 2 * pi * ((l - 1) * (v - 1)/y));
        end
        BB(k, l) = sumRow;
    end
end

F = abs(F);
F = F./(x+y);

% 显示结果
figure;
imshow(F, []);
```

華中科技大學
HUAZHONG UNIVERSITY OF SCIENCE & TECHNOLOGY