作业 2(2022 年 7 月 6 日)

19200300004 黄铭瑞

作业内容

1. 使用最近邻法和双线性插值法对图 1 进行缩放。
2. 对图 1 进行旋转。

（图 1,ford 图标）

作答

1. 首先读入 rgb 图片，并转为灰度图。设定缩放率为 0.7，并通过缩放率对原图大小缩放，创建新的空图片。

   最近邻法：

   Round 四舍五入找到缩放后像素点在原图像的位置。并把原图像该位置的像素值赋给新图像该点处像素。

   双线性插值法：

   计算缩放后像素点坐标对应原图像像素坐标所对应的位置(y, x)，y和x的小数部分即为Δy和Δx. 由Δy和Δx计算出各点权重，

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

   floor()函数对y和x向下取整得到 P1 点坐标像素值。由 P1 点为基点，列出

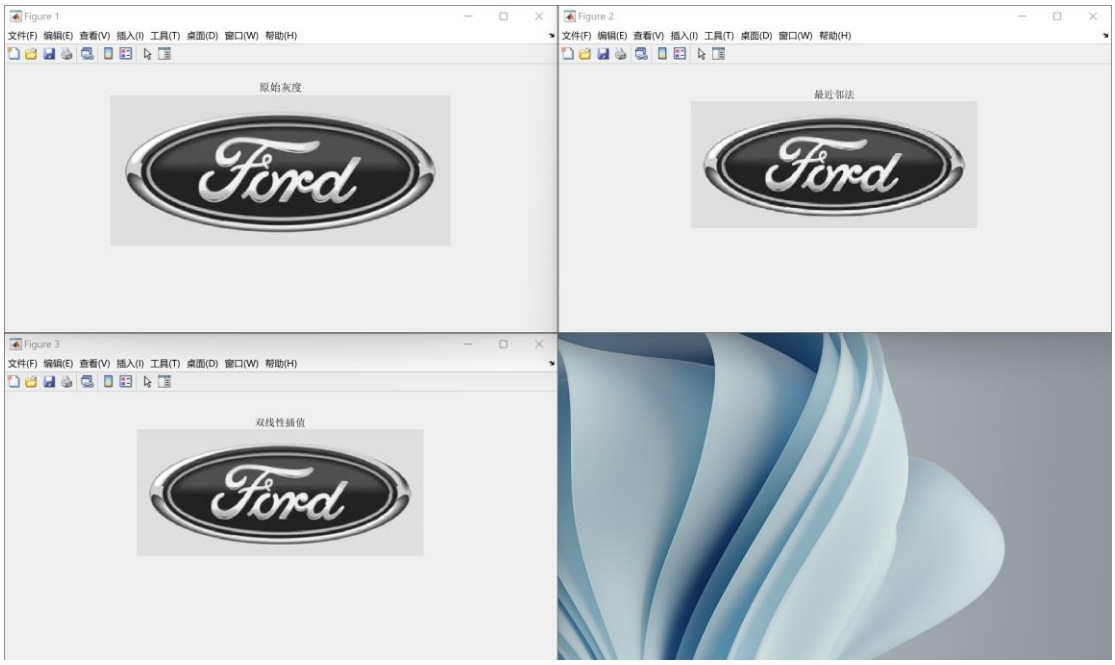
$$P1 = img(P1\_y, P1\_x)$$
$$P2 = img(P1\_y, P1\_x + 1)$$
$$P3 = img(P1\_y + 1, P1\_x)$$

$$P3 = img(P1\_y + 1, P1\_x + 1)$$

由公式

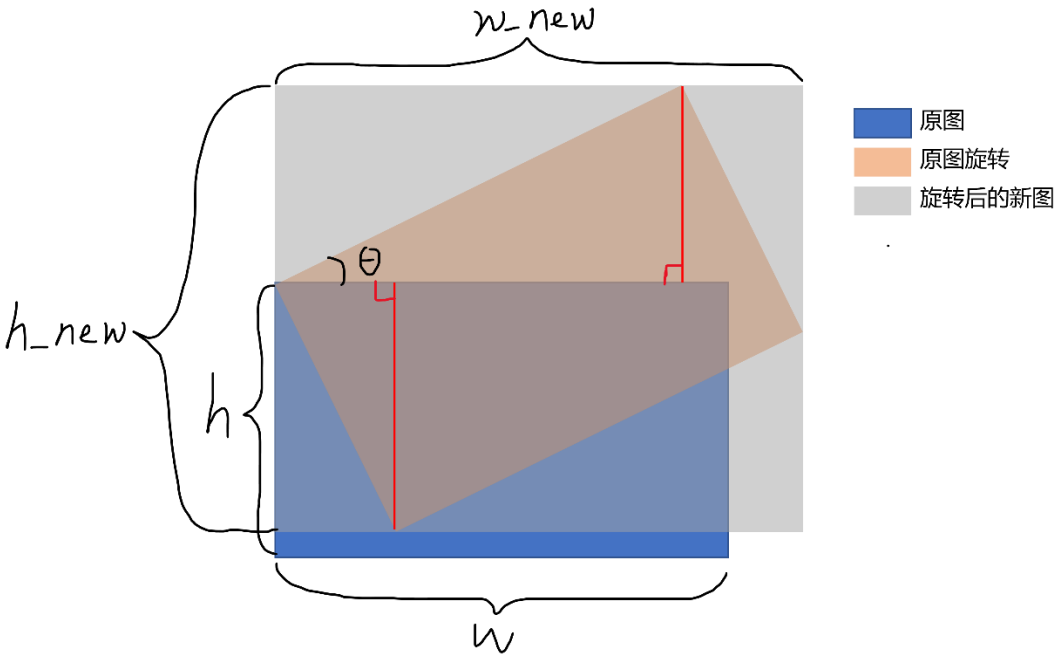
$$P = w1 * P1 + w2 * P2 + w3 * P3 + w4 * P4$$

求出缩放后图像的像素值。缩放结果如图 2 所示，可观察到使用双线性插值的图像更为平滑，最近邻法的图像锯齿较多。



（图 2,原始灰度图，最近邻法，双线性插值法）

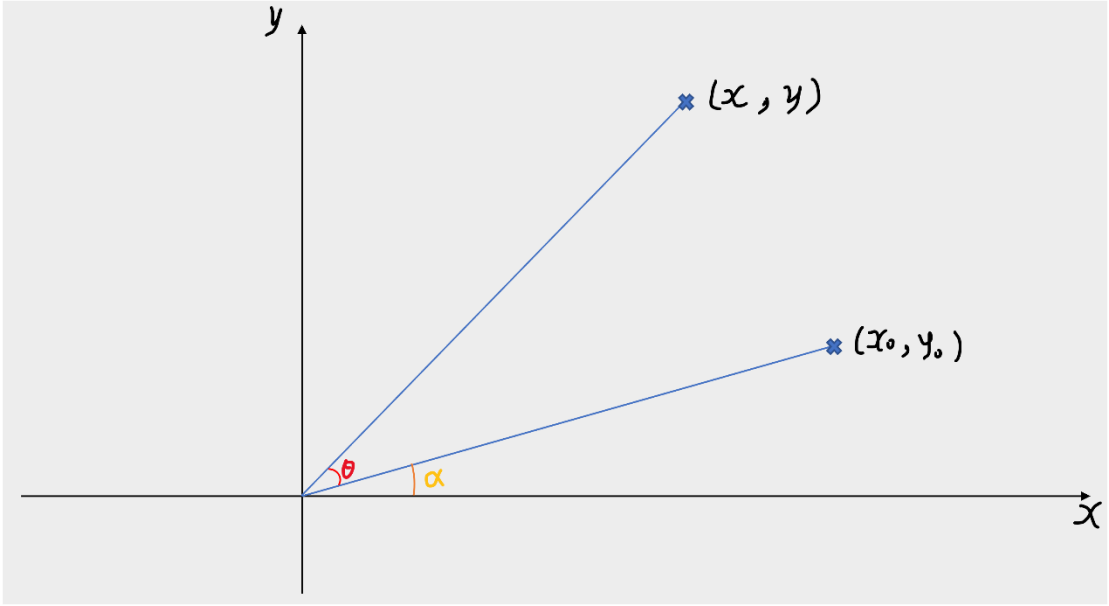
2. 我们先读入图像，根据原图像的高宽和给定的旋转角度大小，计算出旋转后的图像的高宽，并创建新的空图像。新图像高宽计算原理如图 3 所示。



（图 3，图像旋转过程）

$$h_{new} = h * \cos(\theta) + w * \sin(\theta)$$
$$w_{new} = w * \cos(\theta) + h * \sin(\theta)$$

接下来我们计算原图上的点在新图上的位置。一般点的坐标变换如图 4 所示，(x_0, y_0)表示原来的点，(x, y)表示旋转变换后的点。



（图 4，一般点旋转过程）

$$x = R * \cos(\alpha + \theta) = R * (\cos\alpha\, cos\theta - sin\alpha \sin\theta) = x_0 * cos\theta - y_0 sin\theta$$
$$y = R * \sin(\alpha + \theta) = R * (\sin\alpha\, cos\theta + cos\alpha \sin\theta) = y_0 * cos\theta + x_0 sin\theta$$

由于 imread 读取的图像矩阵,是以左上角的点作为原点,出于习惯,把坐标原点转为图像矩形的几何中心,建立像图 4 的坐标系,需要引入坐标系变换矩阵:

$$trans1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w & 0.5h & 1 \end{bmatrix}$$

新坐标系上的旋转矩阵：

$$trans2 = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

转到新图像的图像坐标系的变换矩阵;

$$trans3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w\_new & 0.5h\_new & 1 \end{bmatrix}$$

新图像上的点与旧图像上的点的映射关系矩阵表示:

$$[x \quad y \quad 1] = [x_0 \quad y_0 \quad 1] * trans1 * trans2 * trans3$$

但是该映射会照成新图像上某些点像素值的缺失.需要逆转回原图像进行插值填补缺失的像素值.

逆转矩阵为:

$$inv\_trans1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w\_new & 0.5h\_new & 1 \end{bmatrix}$$

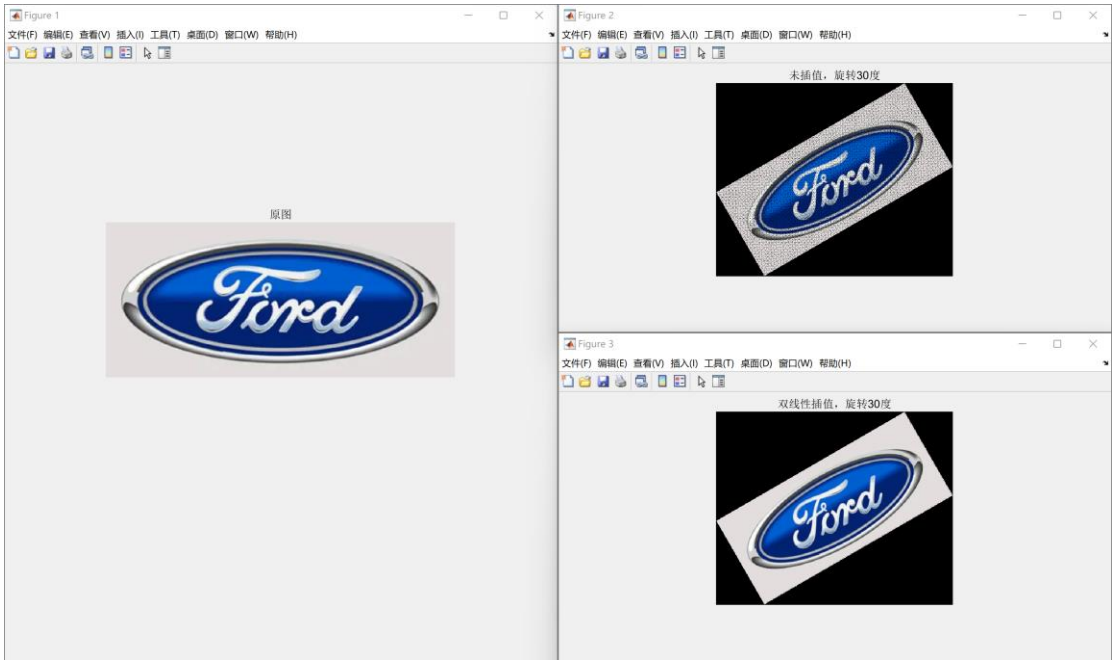
$$inv\_trans2 = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$inv\_trans3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w & 0.5h & 1 \end{bmatrix}$$

逆转关系矩阵表示:

$$[x_0 \quad y_0 \quad 1] = [x \quad y \quad 1] * inv\_trans1 * inv\_trans2 * inv\_trans3$$

接下来在逆转后的图像上使用第一题的双线性插值方法进行插值.实验结果如图5所示。



（图 5，旋转结果）

附录

插值代码：scale.m

```
clc;
clear;
close all;
img = imread('../images/ford.png');
img = rgb2gray(img);
[h, w] = size(img); % 得到原图像的宽高
n = 0.7;
```

```
w_new = floor(w * n); % floor 向下取整
h_new = floor(h * n);
%% 最近邻法
img_near = uint8(zeros(h_new, w_new));
for i = 1: h_new
    for j = 1: w_new
        x = round(j/n); % 缩放后的图像坐标在原图像处的位置
        y = round(i/n);
        img_near(i,j) = img(y,x); %赋值
    end
end
figure;
imshow(img);
title('原始灰度');
figure;
imshow(img_near);
title('最近邻法');
%% 双线性插值法
img_bilinear = uint8(zeros(h_new, w_new));
for i = 1: h_new
    for j = 1: w_new

        x = j/n; % 缩放后的图像坐标在原图像处的位置
        y = i/n;

        % p1 坐标
        p1_x = floor(x);
        p1_y = floor(y);
        % delta
        dx = x - p1_x;
        dy = y - p1_y;
        % 4 邻点权重
        w1 = (1 - dx) * (1 - dy);
        w2 = dx * (1 - dy);
        w3 = (1 - dx) * dy;
        w4 = dx * dy;
        % 获取 4 邻点
        p1 = img(p1_y, p1_x);
        p2 = img(p1_y, p1_x + 1);
        p3 = img(p1_y + 1, p1_x);
        p4 = img(p1_y + 1, p1_x + 1);
        img_bilinear(i, j) = w1 * p1 + w2 * p2 + w3 * p3 + w4 *
p4;%公式
    end
```

```
end
figure;
imshow(img_bilinear);
title('双线性插值');
```

旋转代码：rotate2.m

```
clc;
clear;
close all;
degree = 30;
img = imread('../images/ford.png');
figure;
imshow(img);
title('原图');
[h, w, c] = size(img);
% 新图形的高和宽
h_new = round(h*cosd(degree) + w*sind(degree)) + 1;
w_new = round(w*cosd(degree) + h*sind(degree)) + 1;
% 创建空图像
img_rotate = uint8(zeros(h_new, w_new, c));
% 创建坐标变换矩阵
trans1 = [ 1 0 0; 0 -1 0; -0.5*w 0.5*h 1];
trans2 = [cosd(degree), sind(degree), 0;
          -sind(degree), cosd(degree), 0;
          0, 0, 1];
trans3 = [1 0 0; 0 -1 0; 0.5*w_new 0.5*h_new 1];
% 创建坐标逆变换矩阵
inv_trans1 = [1 0 0; 0 -1 0; -0.5*w_new 0.5*h_new 1];
inv_trans2 = [cosd(degree), -sind(degree), 0;
              sind(degree), cosd(degree), 0;
              0, 0, 1];
inv_trans3 = [1 0 0; 0 -1 0; 0.5*w 0.5*h 1];
%计算每个像素点绕原点旋转后在新图像上的位置。
for j = 1: h
    for i = 1: w
        dst_point = [i j 1]*trans1*trans2*trans3;
        img_rotate(round(dst_point(1,2)),
        round(dst_point(1,1)), :) = img(j, i, :);
    end
end
figure;
imshow(img_rotate);
title('未插值，旋转 30 度');
```

```
for m = 1:h_new
    for n = 1: w_new
        src_point = [n m 1]*inv_trans1*inv_trans2*inv_trans3;
        % 判断该点是否在原图内，在的话进行双线性内插
        if (src_point(1,1)>=1 && src_point(1,1)<=w-1 &&
            src_point(1,2)>=1 && src_point(1,2)<=h-1)
            % img_rotate(m, n) = img(round(src_point(1,2)),
              round(src_point(1,1)));
        % p1 坐标
        p1_x = floor(src_point(1,1));
        p1_y = floor(src_point(1,2));
        % delta
        dx = src_point(1,1) - p1_x;
        dy = src_point(1,2) - p1_y;
        % 4 邻点权重
        w1 = (1 - dx) * (1 - dy);
        w2 = dx * (1 - dy);
        w3 = (1 - dx) * dy;
        w4 = dx * dy;
        % 获取 4 邻点像素值
        p1 = img(p1_y, p1_x, :);
        p2 = img(p1_y, p1_x + 1, :);
        p3 = img(p1_y + 1, p1_x, :);
        p4 = img(p1_y + 1, p1_x + 1, :);
        img_rotate(m, n, :) = w1 * p1 + w2 * p2 + w3 * p3 + w4
        * p4;%公式
        end
    end
end
figure;
imshow(img_rotate);
title('双线性插值，旋转 30 度');
```