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。

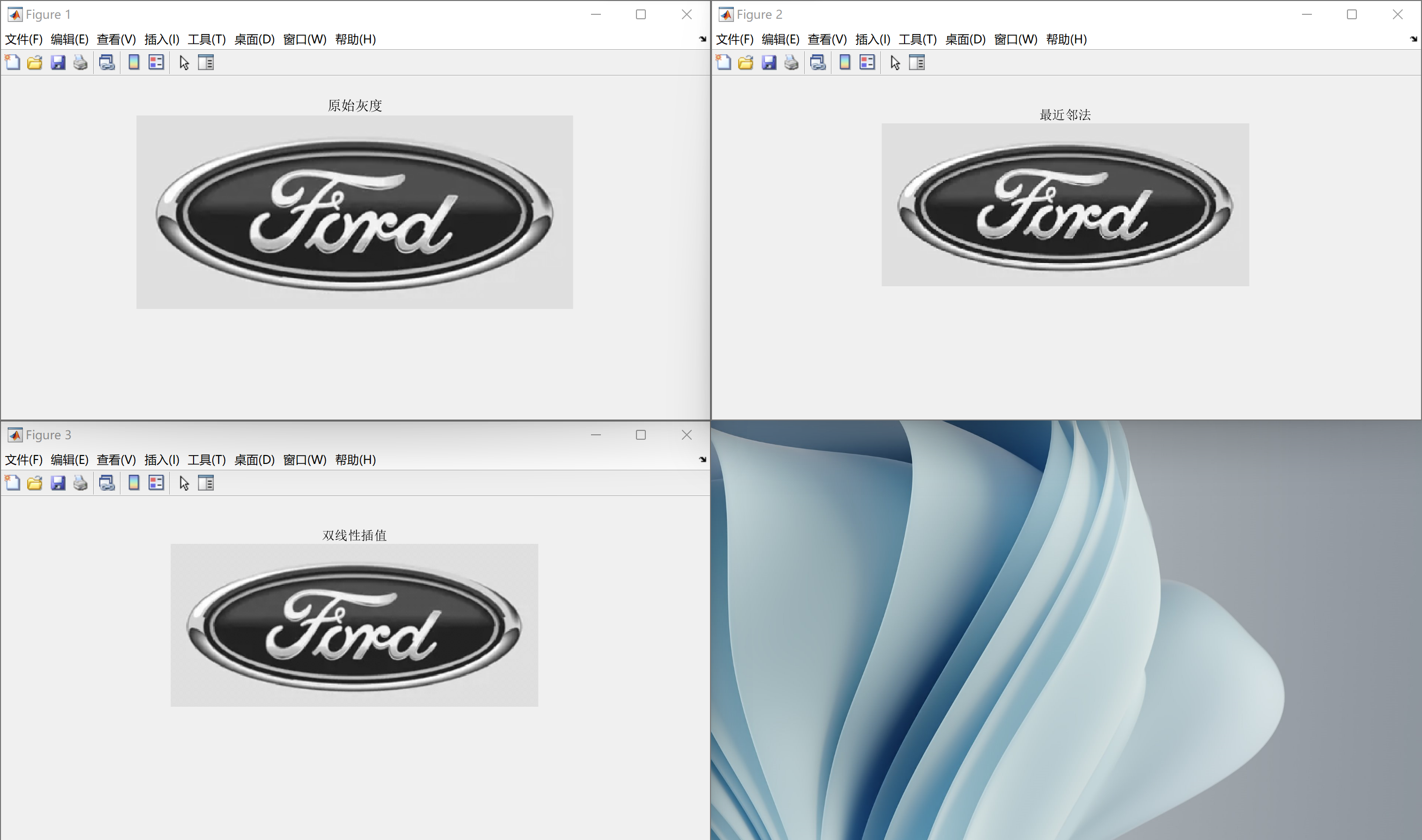
**双线性插值法：**

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,和的小数部分即为和. 由和计算出各点权重，

floor()函数对和向下取整得到P1点坐标像素值。由P1点为基点，列出

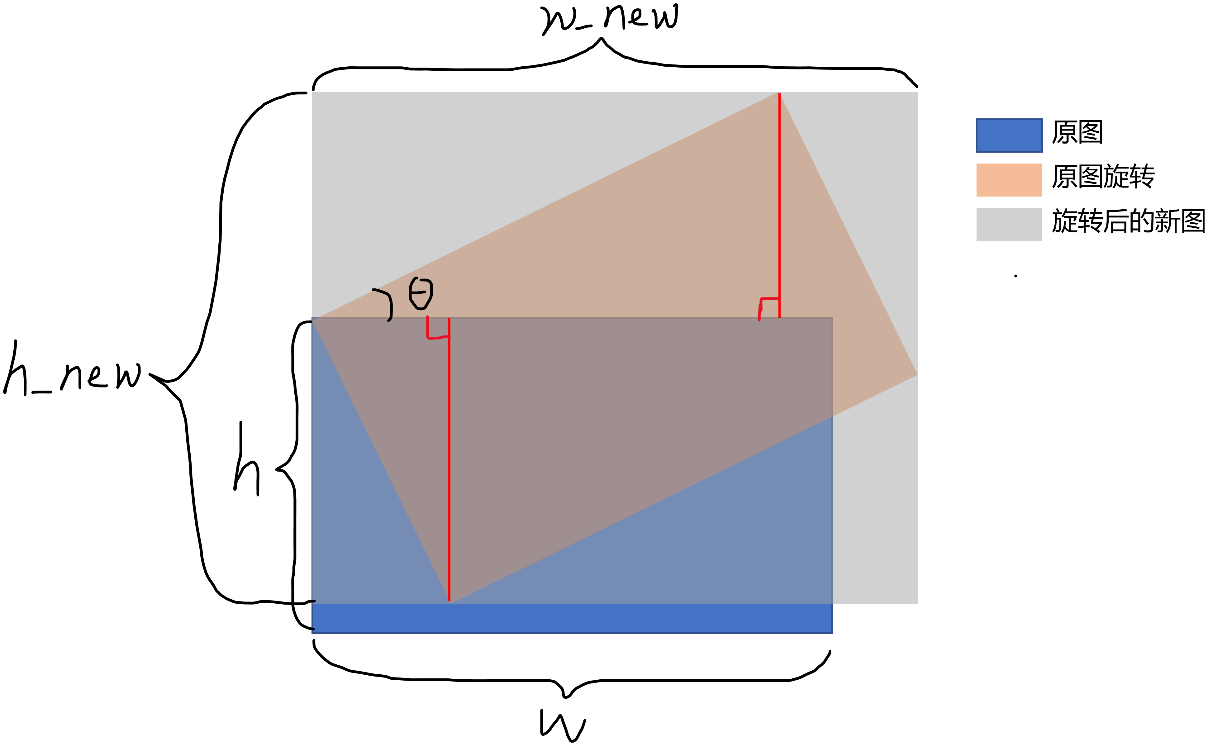
由公式

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。



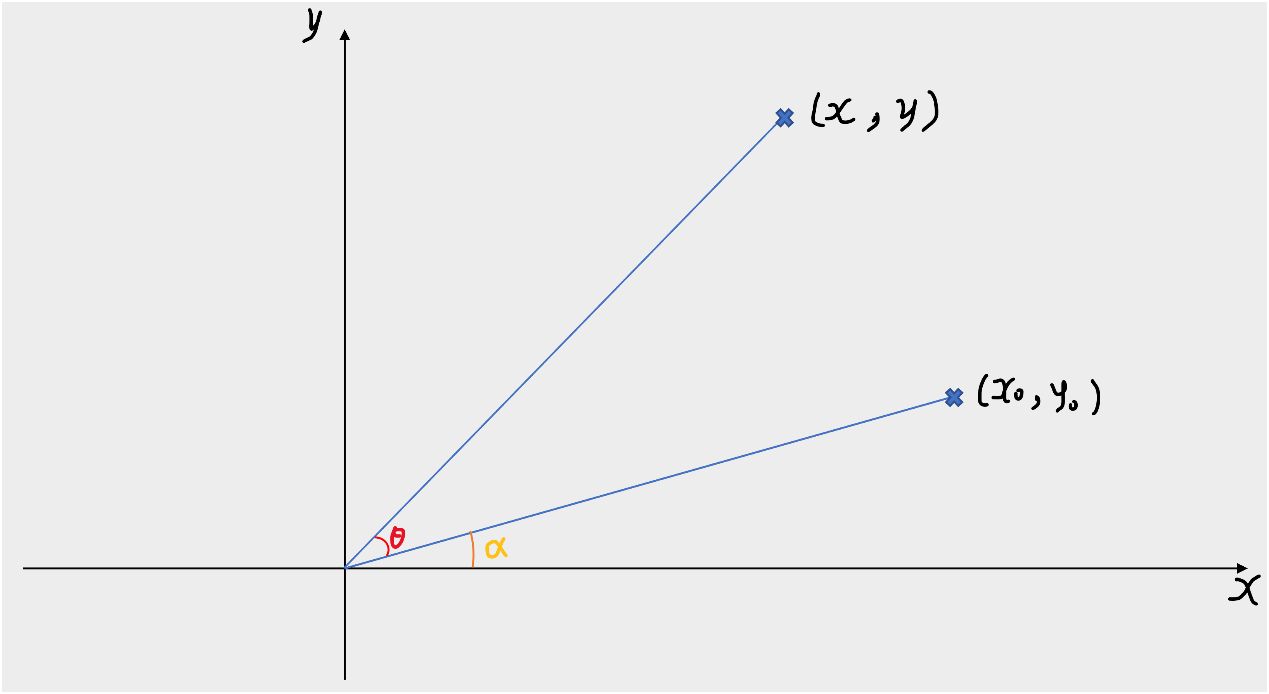
（图2,原始灰度图，最近邻法，双线性插值法）

1. 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）

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，表示原来的点，表示旋转变换后的点。



（图4，一般点旋转过程）

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:

新坐标系上的旋转矩阵：

转到新图像的图像坐标系的变换矩阵;

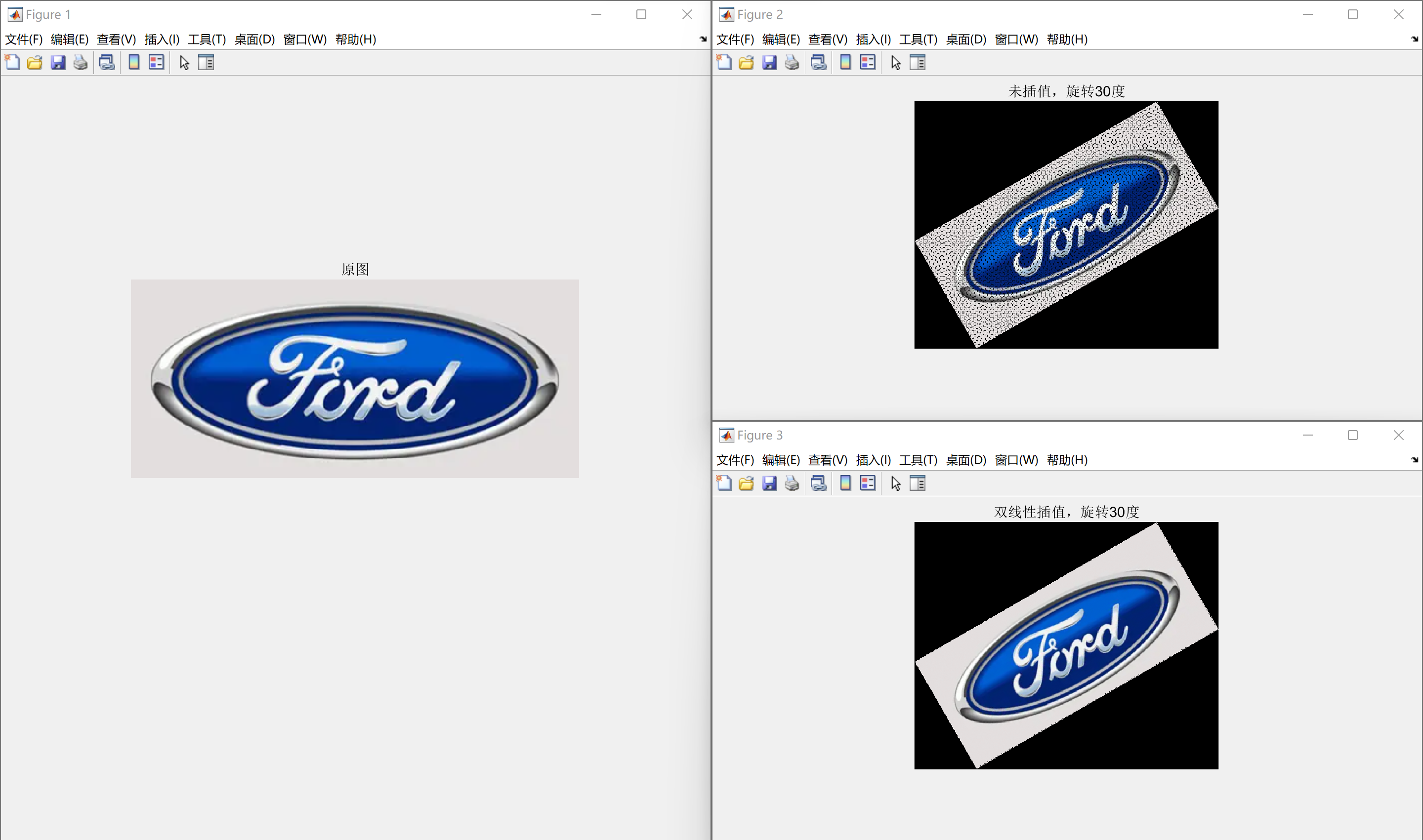
新图像上的点与旧图像上的点的映射关系矩阵表示:

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:

逆转关系矩阵表示:

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');