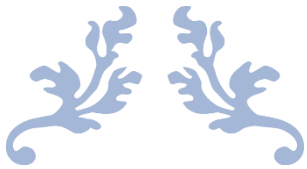


计算机视觉与模式识别

第二次实验报告

王易博

人工智能 82

2186113742

目录

一、实验内容

1、图像的向前变换与逆向变换
2、图像的内插方法
3、图像的几何变换

二、实验原理

2.1 图像的向前变换

将图像坐标的变换方程写出后，转为矩阵格式，直接求出原图像的点所对应的变换后的图像的点。但由于变换后的点的坐标常常是浮点数，所以需要遍历所有点后，才可以通过加权的方式求出每个整数坐标点上的像素值，即变化后的图像，也因此有着空洞的缺点。

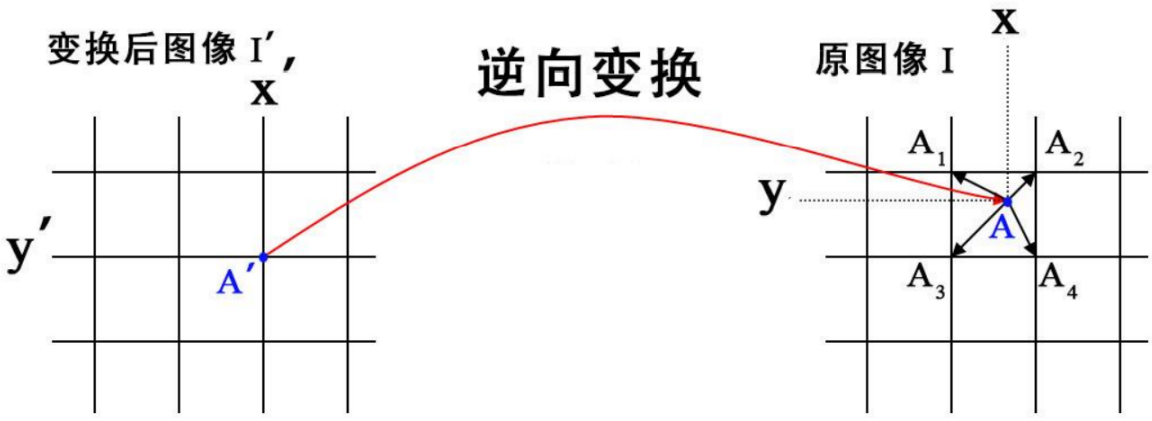
2.2 图像的逆向变换

将图像的转换方程写出后，反解出原图像坐标点的方程，转为矩阵格式，遍历转换后图像的每一点坐标，利用矩阵求出每一点坐标对应的转换前图像的坐标，此时虽然也是浮点数点，但因为原图的每一点坐标是确定的，可以利用插值等方法求出该点的像素值，带回变换后的图像。

2.3 临近插值

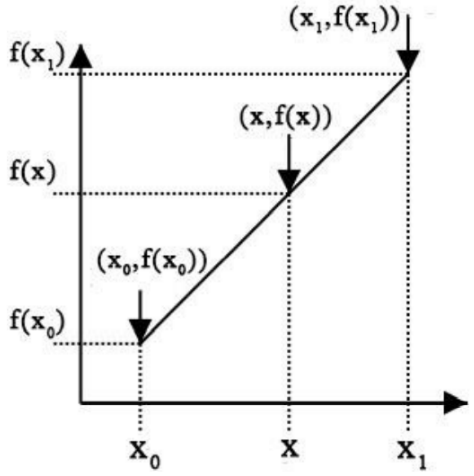
临近插值即在逆向变换后，找到距离浮点数坐标点最近的点的像素值作为变换后图像该店的坐标:

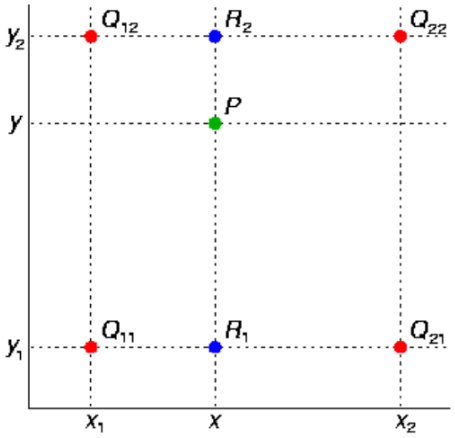
$$I'(A') = I(A_2)$$



2.4 双线性插值

双线性插值即利用浮点数坐标周围四个点的像素值来推断该点的像素，具体是利用浮点坐标与周围四个点的距离来确定插值的比重，对 4 个点的像素值加权求和，得到该点像素值：





$$\frac{f(x)-f(x_0)}{x-x_0}=\frac{f(x_1)-f(x_0)}{x_1-x_0}$$

$$f(x)=f(x_0)+\frac{f(x_1)-f(x_0)}{x_1-x_0}(x-x_0)$$

2.5 平移变换

平移向量 (t_x,t_y)

$$\begin{cases}x'=x+t_x\\y'=y+t_y\end{cases}\Rightarrow\begin{cases}x=x'-t_x\\y=y'-t_y\end{cases}$$

$$\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}1&0&t_x\\0&1&t_y\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}\Rightarrow\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}1&0&-t_x\\0&1&-t_y\\0&0&1\end{bmatrix}\begin{bmatrix}x'\\y'\\1\end{bmatrix}$$

2.6 尺度变换

缩放尺度 (c_x, c_y)

$$\begin{cases} x' = c_x x \\ y' = c_y y \end{cases} \Rightarrow \begin{cases} x = x'/c_x \\ y = y'/c_y \end{cases}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1/c_x & 0 & 0 \\ 0 & 1/c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

2.7 旋转变换

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases} \Rightarrow \begin{cases} x = x'\cos\theta + y'\sin\theta \\ y = -x'\sin\theta + y'\cos\theta \end{cases}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

2.8 欧式变换和相似变换

欧式变换=平移+旋转

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

相似变换=平移+旋转+缩放

$$\begin{bmatrix} c_x\cos\theta & -c_x\sin\theta & t_x \\ c_y\sin\theta & c_y\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2.9 仿射变换

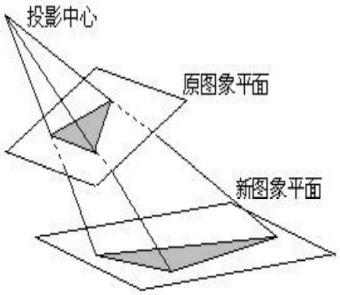
$$\begin{cases} x' = a_{11}x + a_{12}y + t_x \\ y' = a_{21}x + a_{22}y + t_y \end{cases} \Rightarrow \begin{cases} x = \dfrac{1}{a_{11}a_{22} - a_{12}a_{21}}\left(a_{22}x' - a_{12}y' - a_{22}t_x + a_{12}t_y\right) \\ y = \dfrac{1}{a_{11}a_{22} - a_{12}a_{21}}\left(-a_{21}x' + a_{11}y' + a_{21}t_x - a_{11}t_y\right) \end{cases}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} & -a_{22}t_x + a_{12}t_y \\ -a_{21} & a_{11} & a_{21}t_x - a_{11}t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

2.10 射影变换

$$\begin{cases} X = h_{11}x + h_{12}y + h_{13} \\ Y = h_{21}x + h_{22}y + h_{23} \\ Z = h_{31}x + h_{32}y + 1 \end{cases} \Rightarrow \begin{cases} x' = X/Z = \dfrac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1} \\ y' = X/Z = \dfrac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1} \end{cases}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}}_{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow T^{-1}$$

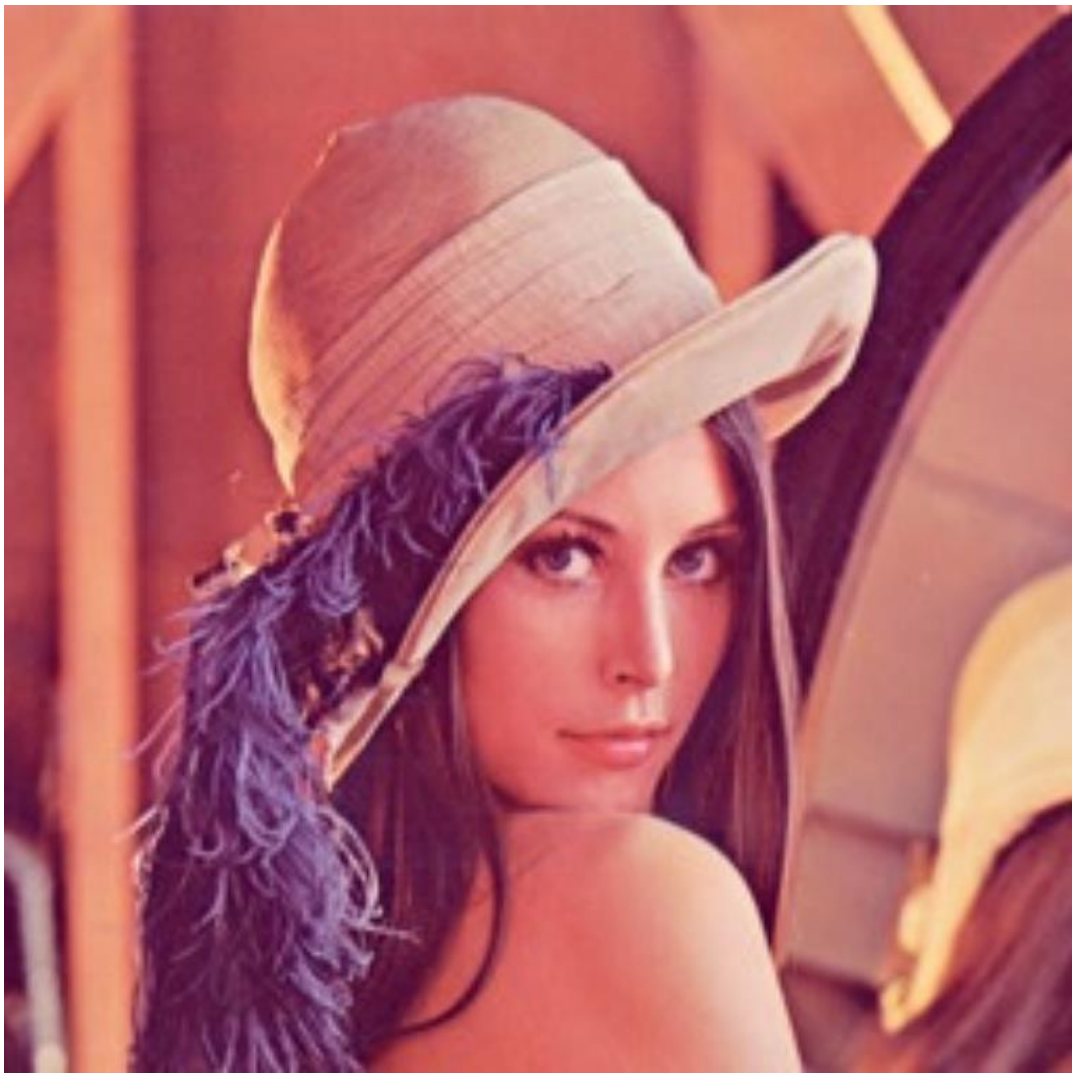


三、实验结果与分析

3.1 临近插值

可以看到，临近插值后的图像锯齿比较明显

3.2 双线性插值



可以看到，比起临近插值，消除了锯齿

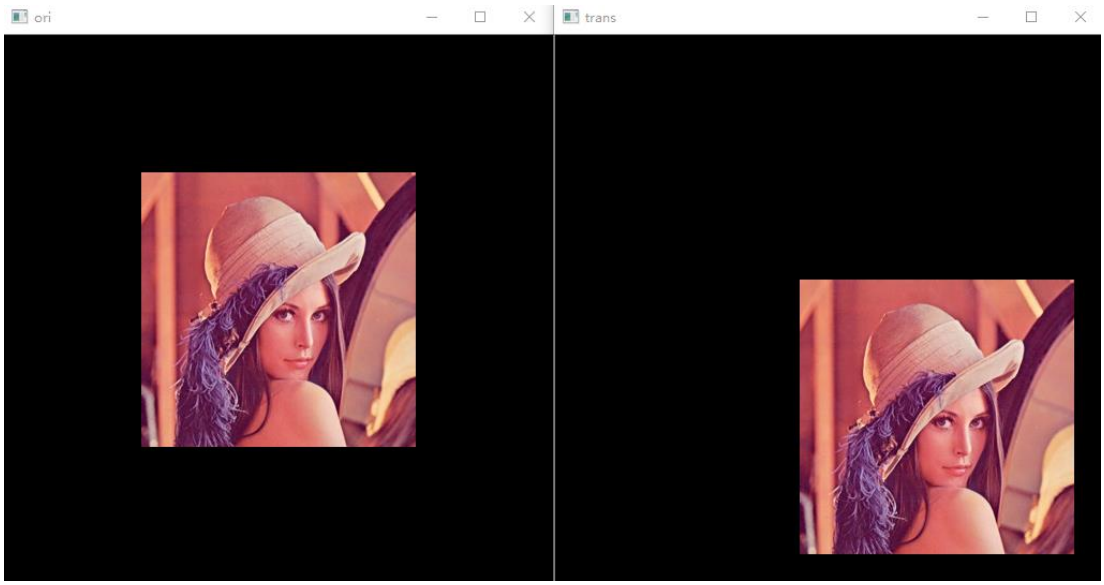
3.3 平移变换

平移矩阵如下所示:

```
tx, ty = 100, 100
```

```
[[1, 0, -tx],
 [0, 1, -ty],
 [0, 0, 1]],
```

变换结果:



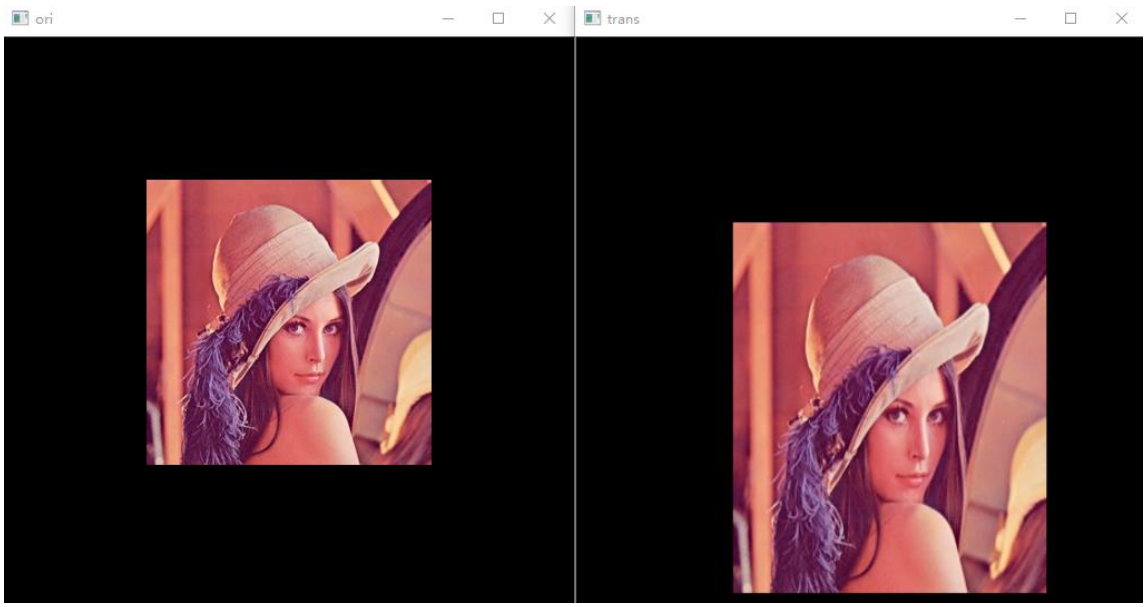
3.4 尺度变换

变换矩阵如下:

```
cx, cy = 1.1, 1.3
```

```
[[1 / cx, 0, 0],
 [0, 1 / cy, 0],
 [0, 0, 1]],
```

变换结果:



3.5 旋转变换

在旋转变换时，由于方程是针对直角坐标系的，即原点在中央，而图像的下标是从左上角开始的，所以旋转中心会在左上角，为了解决这个问题，在逆向映射之前，先将坐标原点移动到图像中央，然后进行逆向映射求出像素值，最后再将坐标变换图像坐标赋值即可，坐标转换代码如下：

```
if shift:
    bias_x = img.shape[1] // 2
    bias_y = img.shape[0] // 2
    coordinate[0, :] -= bias_x
    coordinate[1, :] = -coordinate[1, :] + bias_y
```

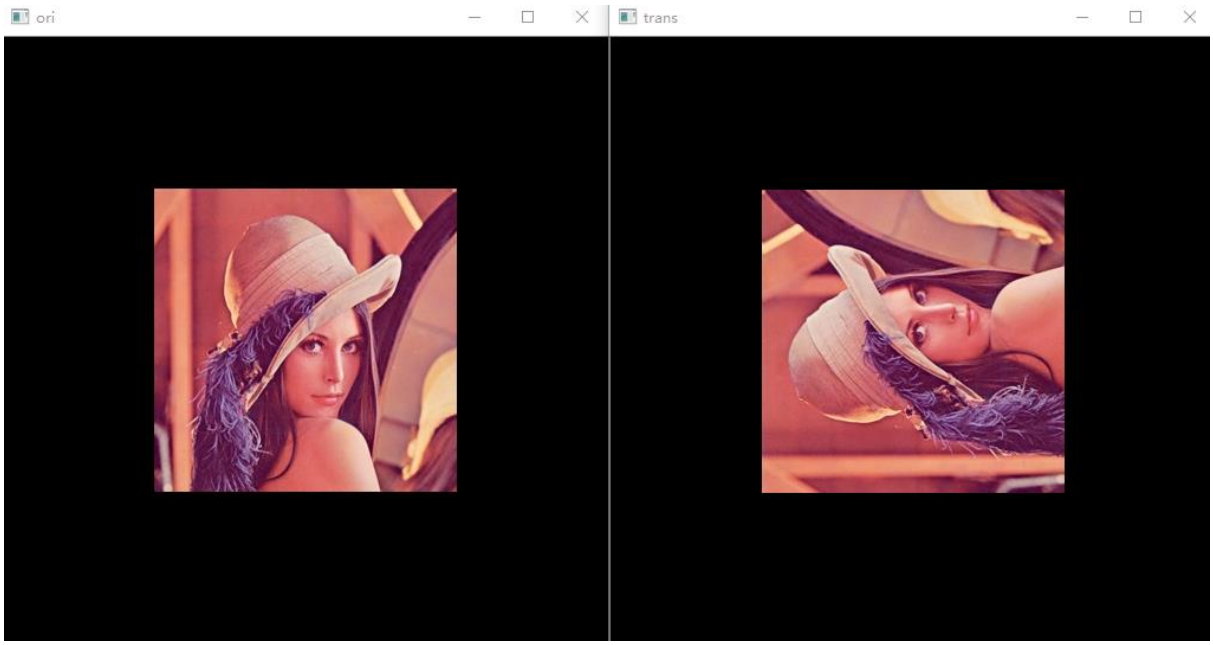
```
if shift:
    x_ori += bias_x
    x += bias_x
    y_ori = -y_ori + bias_y
    y = -y + bias_y
```

变换矩阵：

```
theta = np.pi / 2
```

```
[[np.cos(theta), np.sin(theta), 0],
 [-np.sin(theta), np.cos(theta), 0],
 [0, 0, 1]],
```

变换结果：

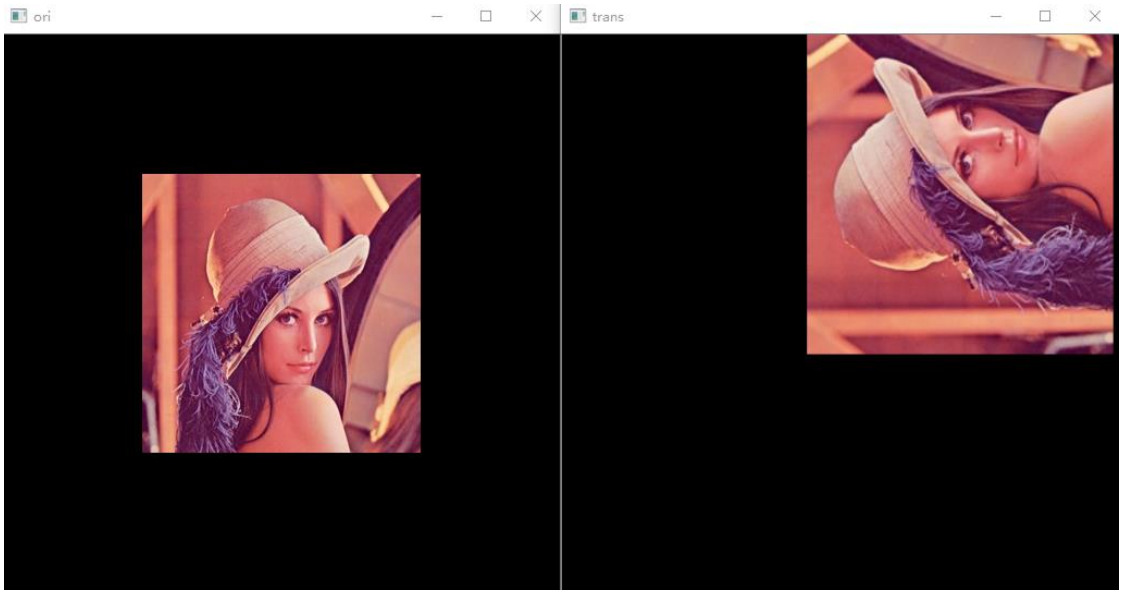


3.6 欧式变换与相似变换

该变换即将前三种变换融合，可以直接对变换矩阵进行相乘，然后图像会按照相乘顺序进行变换，操作如下图所示：

```
trans(img_zero, matrix_trans[2] @ matrix_trans[0] @ matrix_trans[1], shift=True)
```

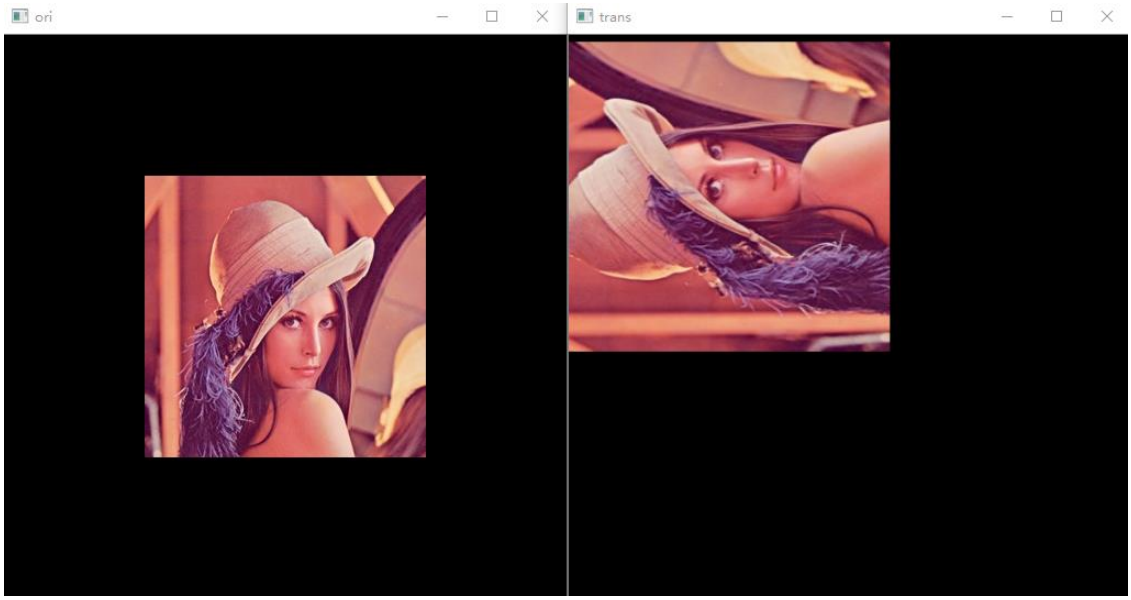
该操作为先绕原点进行旋转，然后再进行平移、缩放变换，结果如图所示：



改变矩阵相乘顺序：

```
trans(img_zero, matrix_trans[0] @ matrix_trans[1] @ matrix_trans[2], shift=True)
```

该操作为先平移、缩放变换，然后再绕原点进行旋转，结果如图所示：



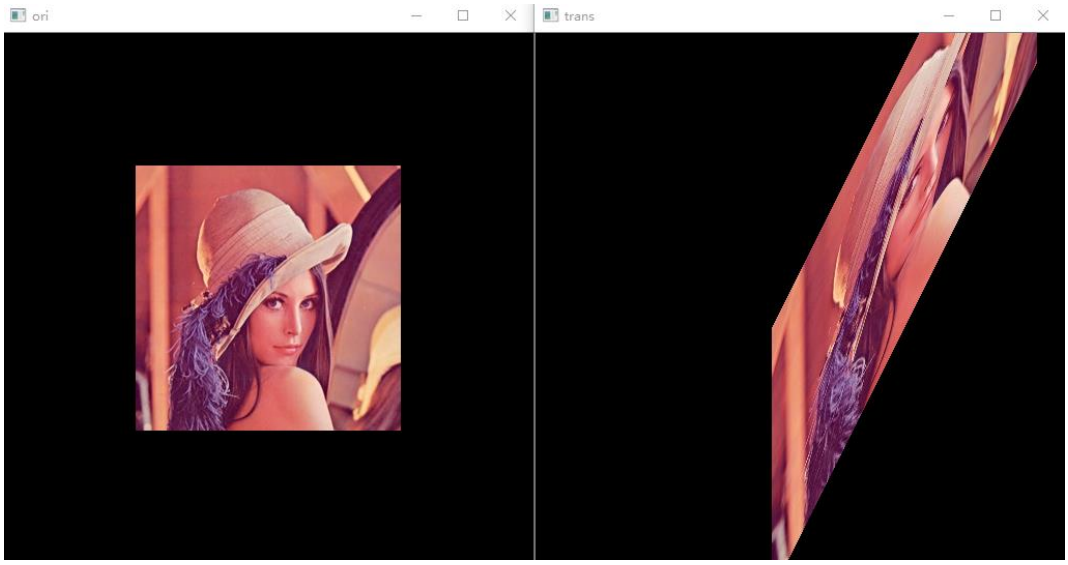
3.7 仿射变换

变换矩阵如图所示：

```
a_11, a_12, a_21, a_22 = 1, 0, 2, 1
coefficient = a_11 * a_22 - a_12 * a_21
```

```
np.multiply([[a_22, -a_12, -a_22 * tx + a_12 * ty],
             [-a_21, a_11, a_21 * tx - a_11 * ty],
             [0, 0, 1]], 1 / coefficient),
```

变换结果：



此图与 ppt 所给不同的原因是此处采用的是直角坐标系，y 的坐标和图片的坐标相反，所以变换后上下相反

3.8 射影变换

由于射影变换直接求逆矩阵后，会求出 Z 值，若要转换到二维，则需将 Z 归为 1：

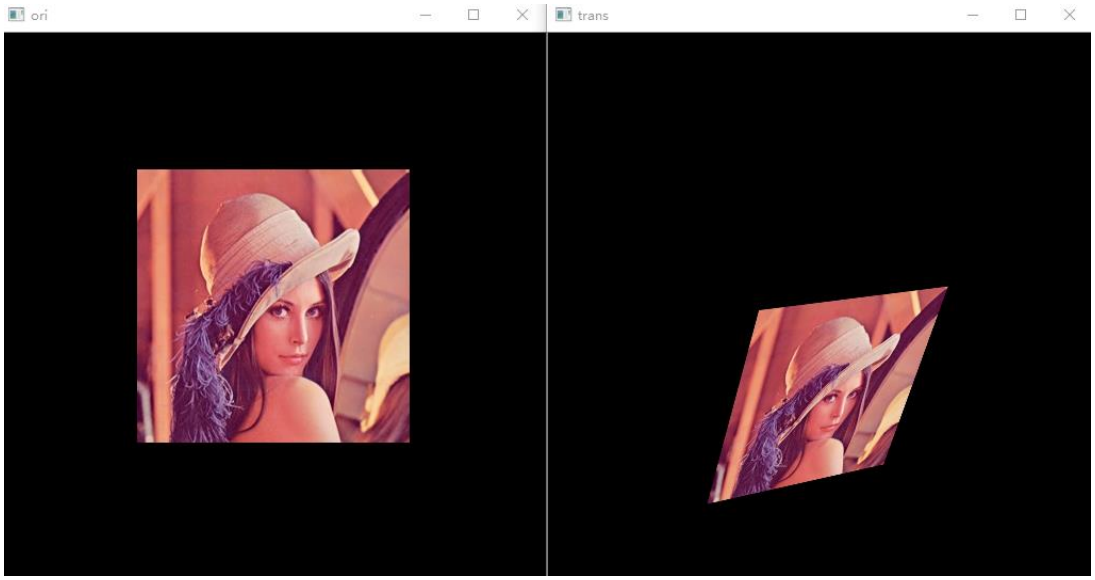
```
coordinate_ori /= coordinate_ori[2, :]
```

变换矩阵如下：

```
h_11, h_12, h_13, h_21, h_22, h_23, h_31, h_32 = 0.97, -0.14, 117, 0.02, 1.02, 160, 0.0005, 0.0005
```

```
np.linalg.inv([[h_11, h_12, h_13],
               [h_21, h_22, h_23],
               [h_31, h_32, 1]])
```

变换结果：



四、结论与讨论

4.1 图像插值

由于邻近插值是选取周围某点的值，这种非线性的插值会造成像素值的“不连续”，所以会有很强的锯齿感，而双线性插值使得图像像素“连续”，所以可以消除锯齿感

4.2 平移、缩放、旋转、相似、仿射变换

这五个变换本质上都是二维到二维的映射，区别在于用于映射的矩阵的数值

4.3 射影变换

该变换不同于以上五种变换，是从三维到二维的映射，因此再用逆向变换时，需要将变回的坐标除 Z 的值，使得 Z 归一化，即映射到二维平面。

五、代码

https://github.com/wyb2333/Computer-Vision-and-Pattern-Recognition