# 图像透视变换

## 一、 实验目的

读入一幅图像（可以是灰度图像也可以是多通道图像），通过透视变换得到新的图像，并将得到的图像和原图像对比输出。

## 二、 实验环境

编程语言：Python3.7

操作系统：Windows10

代码编辑器：vscode

调用 Python 库：sys、numpy、matplotlib

## 三、 算法原理

给定图像，先将该图像放入到三维空间中，得到每个像素的对应坐标值$\mathrm{P}=(\mathrm{x}_0,\mathrm{y}_0,\mathrm{z}_0)$，接着指定相机位置$\mathrm{V}=(\mathrm{v_x},\mathrm{v_y},\mathrm{v_z})$、裁剪平面的法向量$\mathrm{N}=(\mathrm{n_x},\mathrm{n_y},\mathrm{n_z})$，那么根据透视原理，推导得到图像投影在裁剪平面上新坐标为$P'=(\mathrm{k}(\mathrm{v_x}-\mathrm{x}_0)+\mathrm{x}_0,\mathrm{k}(\mathrm{v_y}-\mathrm{y}_0)+\mathrm{y}_0,\mathrm{k}(\mathrm{v_z}-\mathrm{z}_0)+\mathrm{z}_0)$，其中$k=1+\frac{1}{N\cdot(V-P)}$。

由于最终需要二维坐标，因此还要将$P'$，转化为二维形式$P''$，具体做法如下：首先用原图像第一行第一列的像素点坐标$\mathrm{P}_0$（投影点为$\mathrm{P}_0'$）作为基准，接着任取原图像水平方向的两个像素点的坐标$\mathrm{P}_1$和$\mathrm{P}_2$（对应的投影点分别为$\mathrm{P}_1'$和$\mathrm{P}_2'$），用向量$\mathrm{P}_1'\mathrm{P}_2'$作为二维坐标的x轴的投影方向，进而向量$\mathrm{P}_0'P'$在$\mathrm{P}_1'\mathrm{P}_2'$的投影长度$\frac{P_0'P'\cdot P_1'P_2'}{|P_0'P'|}$即为新的x坐标。同样的，取竖直方向上的两个像素点的坐标$\mathrm{P}_3$和$\mathrm{P}_4$，得到新的y坐标$\frac{P_0'P'\cdot P_3'P_4'}{|P_0'P'|}$。最终$P''=\left(\frac{P_0'P'\cdot P_1'P_2'}{|P_0'P'|},\frac{P_0'P'\cdot P_3'P_4'}{|P_0'P'|}\right)$。

为了方便计算，原像素点坐标P的$z_0$取为$0$，$x_0$和$y_0$直接用像素点所在行数和列数来表示。

## 四、 代码流程

**主流程：**

1、 根据图像的相对路径读入图片

```
img = plt.imread(sys.path[0] + '/../res/1.jpg')
```

2、 调用透视变换函数

```
# 透视变换，参数包含图像、裁剪平面的法向以及相机位置
new_img = transform(img, (-1, -1, -5), (500, 500, 500))
```

3、 最后将变换前后的图像对比显示

```
axes1 = plt.subplot(121)
axes1.imshow(img, cmap='gray')
axes2 = plt.subplot(122)
axes2.imshow(new_img, cmap='gray')
plt.show()
```

**投影变换函数：**

1、 找到原图像像素点坐标对应的新坐标，并将新坐标保存在 tran 矩阵中

```
trans = np.empty((img.shape[0], img.shape[1], 3), np.float)
for x in range(trans.shape[0]):
    for y in range(trans.shape[1]):
        origin = np.array((x, y, 0), np.int16)
```

```
        k = 1 + 1 / np.dot(normal_vec, view_point - origin)
        trans[x][y] = k * (view_point - origin) + origin
```

2、 将得到的三维坐标转化为图像的二维坐标，并根据坐标的范围做一定的缩放处理

```
trans -= trans[0, 0]
axisx = trans[-1, 0] / np.sqrt(trans[-1, 0].dot(trans[-1, 0]))
axisy = trans[0, -1] / np.sqrt(trans[0, -1].dot(trans[0, -1]))
trans[:, :, 0], trans[:, :, 1] = trans.dot(axisx), trans.dot(axisy)
min_x, min_y, min_z = np.min(trans[(0, 0, -1, -1), (0, -1, 0, -1)], axis=0)
max_x, max_y, _ = np.max(trans[(0, 0, -1, -1), (0, -1, 0, -1)], axis=0)
ratio = img.shape[0] / (max_x - min_x)
trans = np.array((trans - [min_x, min_y, min_z]) * ratio, np.int16)
height = np.int16((max_x - min_x) * ratio)
width = np.int16((max_y - min_y) * ratio)
```

3、 最后根据新的二维坐标填充像素点，得到新的图像（这里根据具体是灰度图像还是多通道图像做了特殊处理）

```
if len(img.shape) == 2:    # 灰度图像
    new_img = np.zeros((height + 1, width + 1), dtype=np.uint8)
else:    # 多通道图像
    new_img = np.zeros((height + 1, width + 1, img.shape[2]), dtype=np.uint8)


for x in range(trans.shape[0]):
    for y in range(trans.shape[1]):
        new_img[tuple(trans[x, y, (0, 1)])] = img[x, y]
```

## 五、 实验结果

选取不同的裁剪平面法向量和相机位置可以得到不同的透视变换结果，而且同时支持灰度图像和多通道图像。

Figure 1 灰度图的投影变换

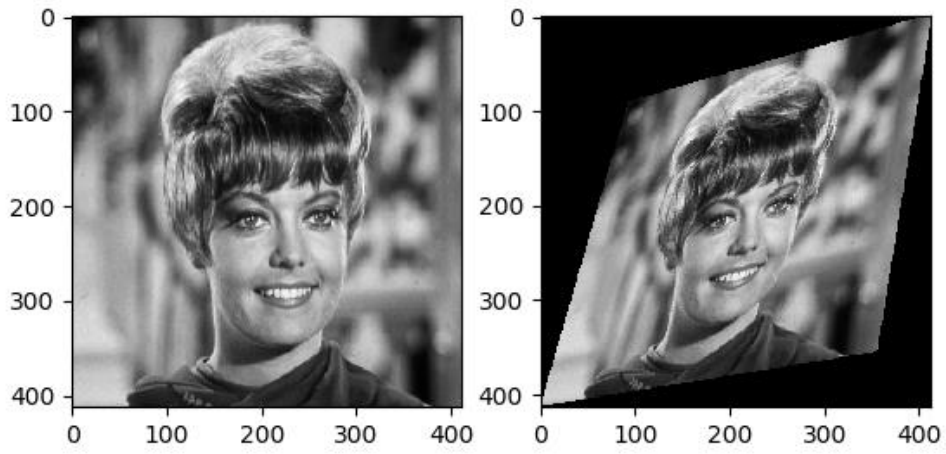


Figure 2 取定裁剪平面法向量为(-1,-1,-5)以及相机位置(500,500,500)时得到的投影变换图像

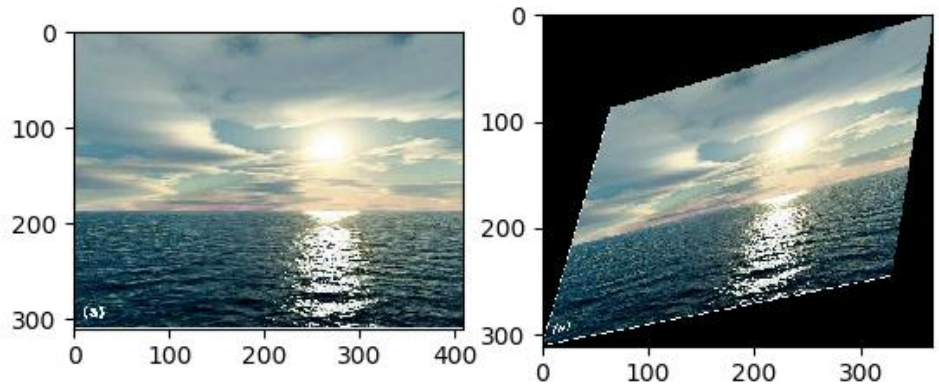
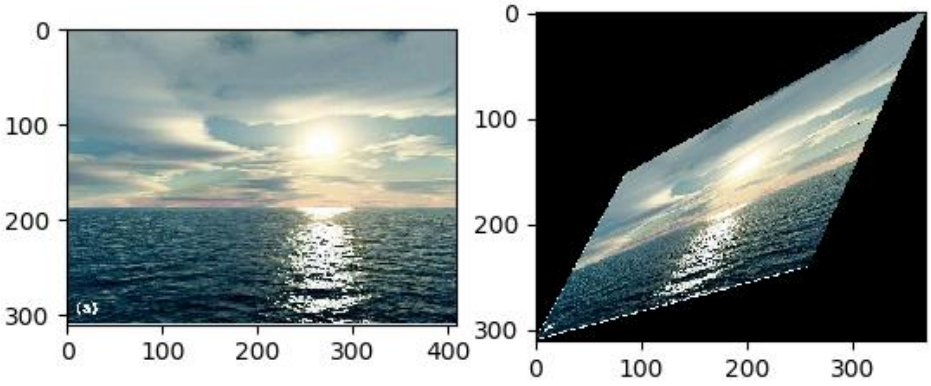



Figure 3 取定裁剪平面法向量为(-1,-5,-5)以及相机位置(500,500,500)时得到的投影变换图像




## 六、 总结分析

通过图像的透视变换了解了基本的图像存储结构、操作方法以及不同类型图像之间的差异。透视变换涉及到了像素点的位置变换，是我们对数字图像处理的非常基础的一步，通过该实验掌握了基本的透视原理和变换方法。