计算机视觉与应用实践实验报告（四）

目录

计算机视觉与应用实践实验报告（四）.....1
一、 实验目的.....1
二、 实验原理.....1
三、 实验步骤.....2
四、 关键程序代码.....2
五、 实验结果.....3
六、 实验分析与总结.....4

一、实验目的

- 单应性变换，计算图片之间的单应性变换，需要详细的实验过程和结果分析。单应性变换 (Homography) 是将一个平面内的点映射到另一个平面内的二维投影变换。

二、实验原理

2.1 单应性变换

单应性变换，可简单理解为用来描述物体在世界坐标系和像素坐标系之间的位置映射关系，对应的变换矩阵称为单应性矩阵。单应性在图像校正、图像拼接、相机位姿估计、视觉 SLAM 等领域有非常重要的作用。

如何估计单应矩阵？

首先，假设两张图像中的对应点对对齐次坐标为（x',y',1）和（x,y,1），单应矩阵 H 定义为：

$$H=\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

则有：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

矩阵展开后有 3 个等式，将第 3 个等式代入前两个等式中可得：

$$x'=\frac{h_{11}x+h_{12}y+h_{13}}{h_{31}x+h_{32}y+h_{33}}$$

$$y'=\frac{h_{21}x+h_{22}y+h_{23}}{h_{31}x+h_{32}y+h_{33}}$$

也就是说，一个点对对应两个等式。

三、实验步骤

四、关键程序代码

1、实现鼠标点击获取像素坐标（目标图像的获取方式相同，不再重复展示）

```
# mouse callback function
def select_points_src(event,x,y,flags,param):
    global src_x, src_y, drawing
    if event == cv.EVENT_LBUTTONDOWN:
        drawing = True
        src_x, src_y = x,y
        cv.circle(src_copy,(x,y),5,(0,0,255),-1)
    elif event == cv.EVENT_LBUTTONUP:
        drawing = False
```

2、 求单应性矩阵并进行单应性变换

```
def get_plan_view(src, dst):
    src_pts = np.array(src_list).reshape(-1,1,2)
    dst_pts = np.array(dst_list).reshape(-1,1,2)
    H, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC,5.0)
    print("H:")
    print(H)
    plan_view = cv.warpPerspective(src, H, (dst.shape[1], dst.shape[0]))
    return plan_view
```

3、 总体框架逻辑（通过鼠标交互的方式得到两张图中对应的 corresponding Points，进而计算单应性矩阵）

```
while(1):
    cv.imshow('src',src_copy)
    cv.imshow('dst',dst_copy)
    k = cv.waitKey(1) & 0xFF
    if k == ord('s'):
        print('save points')
        cv.circle(src_copy,(src_x,src_y),5,(0,255,0),-1)
        cv.circle(dst_copy,(dst_x,dst_y),5,(0,255,0),-1)
```

```
        src_list.append([src_x,src_y])
        dst_list.append([dst_x,dst_y])
        print("src points:")
        print(src_list);
        print("dst points:")
        print(dst_list);
    elif k == ord('h'):
        print('create plan view')
        plan_view = get_plan_view(src, dst)
        cv.imshow("plan view", plan_view)
```

五、实验结果

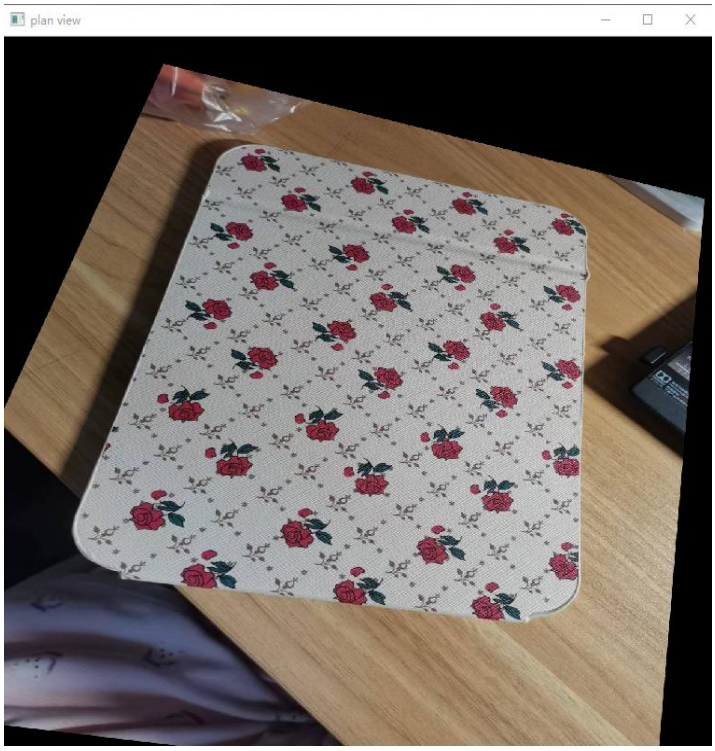
（1）原图片和目标图片对应 corresponding Points 展示（图中绿色的点），左为原图，右为目标图。

坐标点以及变化矩阵输出：

```
save points
src points:
[[201, 218], [965, 175], [193, 1334], [1072, 1301]]
dst points:
[[414, 259], [1022, 430], [126, 1201], [1016, 1354]]
```

```
create plan view
H:
[[ 8.96296827e-01 -2.77769707e-01  2.92835915e+02]
 [ 3.10370807e-01  6.16153007e-01  6.53545020e+01]
 [ 1.04333394e-04 -1.78197800e-04  1.00000000e+00]]
```

（2）对原图进行单应性变换之后的结果



六、实验分析与总结

本次实验主要是对目标物体（折叠镜）为基准在两图之间进行单应性转化，将原图中的镜子转换成和目标图片中目标物体相同的角度和方向，单应性变换将一个图像中的点映射到另一个图像中的相应点，他反应了从一个平面到另一个平面的映射关系。相较于 TPS 等非刚性变换来说算是空间几何变化中较为简单的变换方式。

我们都知道，单应性变换矩阵是一个 3*3 的矩阵 H。这个变换可以被任意乘上一个非零常数，而不改变变换本身。所以它虽然具有 9 个元素，但是具有 8 个自由度。这意味这它里面有 8 个未知参数待求；矩阵中[3,3]位置的变量值为 1，正如我们实验中计算出的 H 那样。

和其他几何变换相比，单应变换具有 8 个自由度，其它一些更简单的变换也是 3*3 矩阵，只不过它们包含一些特定的约束来降低自由度的数量。从紧约束至松约束，逐步接近单应变换。

单应性在图像校正、图像拼接、相机位姿估计、视觉 SLAM 等领域有非常重要的作用。