计算机视觉实践实验报告

目录

一. 实验目的

- 熟悉单应性变换原理。
- 对单应性变换进行实验，计算图片之间的单应性变换，并进行结果分析。

二. 实验原理

单应性原理被广泛应用于图像配准，全景拼接，机器人定位SLAM，AR增强现实等领域。

2.1 单应性变换

单应性变换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。在这里，平面是指图像或者三维中的平面表示。单应性变换具有很强的实用性，比如图像配准，图像纠正和纹理扭曲，以及创建全景图像，我们将频繁的使用单应性变换。本质上，单应性变换H，按照下面的方程映射二维中的点（齐次坐标意义下）：

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

或者

$$x' = Hx$$

对于图像平面内（甚至是三维中的点，后面我们会介绍到）的点，齐次坐标是个非常有用的表示方式。点的齐次坐标是依赖于其尺度定义的，所以，x=[x,y,w]=[ax,ay,aw]=[x/w,y/w,1]都表示同一个二维点。因此，单应性矩阵H也仅依赖尺度定义，所以，单应性矩阵具有8个独立的自由度。我们通常使用w=1来归一化点，这样，点具有唯一的图像坐标x和y。这个额外的坐标是的我们可以简单地使用一个矩阵来表示变换。

2.2 单应性矩阵的理解及直接线性变换算法

在齐次坐标中，假设一点p(xi,yi,1)经过H矩阵的变换变为p‘(xi’,yi’,1)，即 p’ = Hp，通常，对于直接线性变换，一个完全射影变换具有8个自由度，H矩阵有8个自由度，这样至少需要

4对特征点对求解。4个特征点对可以建立8个方程。那么对于有n对特征点的情况(超定方程)，解p’= Hp方程组可以转化为对齐次方程组Ax = 0的求解。而对Ax = 0的求解转化为 min ||Ax||2 的非线性优化问题（超定方程，通过最小二乘拟合得到近似解）。

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\mathbf{x'} \qquad \mathbf{H} \qquad \mathbf{x}$$

进一步可得：

$$\begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

这样便可构造系数矩阵：

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix}$$

通过系数矩阵我们可以构造出齐次线性方程组(Ax = 0)：

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

即：

Direct Linear Transforms

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

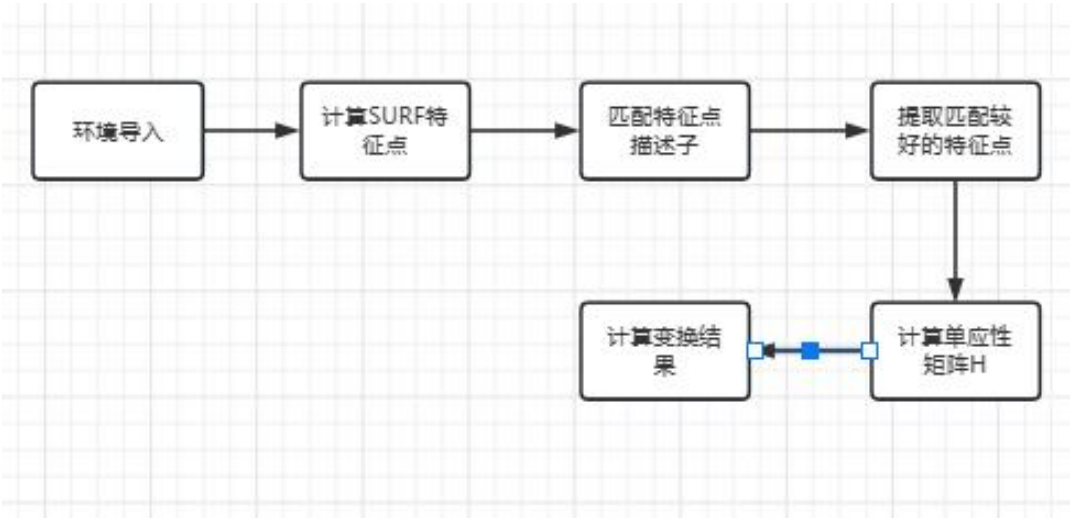
$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

$$\text{minimize } \|\mathbf{Ah} - \mathbf{0}\|^2$$

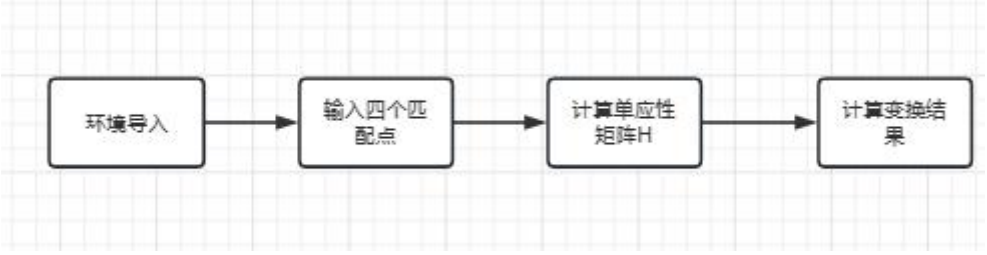
对于这样的超定方程求解，可以通过最小二乘的方式求解。通过对系数矩阵A求取特征值和特征向量得到。通过以下方式获得最小二乘解：[V,D] = eig(A'*A) ，其中D是特征值对角矩阵(特征值沿主对角线降序)，V是对应D特征值的特征向量(列向量)组成的特征矩阵,A'表示A的转置。其最小二乘解为V(1)，即系数矩阵A最小特征值对应的特征向量就是超定方程组Ax = 0的最小二乘解，至此，H矩阵已经求取。

三. 实验步骤

传统方法：



使用opencv中的单应性变换函数：



四. 程序代码

- 读取图片，计算SURF特征点和对应的描述子，kp存储特征点坐标，des存储对应描述子。

```
# 读取图片
im1 = cv2.imread('left.png')
im2 = cv2.imread('right.png')

# 计算SURF特征点和对应的描述子，kp存储特征点坐标，des存储对应描述子
surf = cv2.xfeatures2d.SURF_create()
kp1, des1 = surf.detectAndCompute(im1, None)
kp2, des2 = surf.detectAndCompute(im2, None)
```

- 通过特征点坐标计算单应性矩阵H，findHomography中使用了RANSAC算法剔除错误匹配。

```
# 匹配特征点描述子
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

# 提取匹配较好的特征点
good = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        good.append(m)
```

- 使用单应性矩阵计算变换结果并绘图。

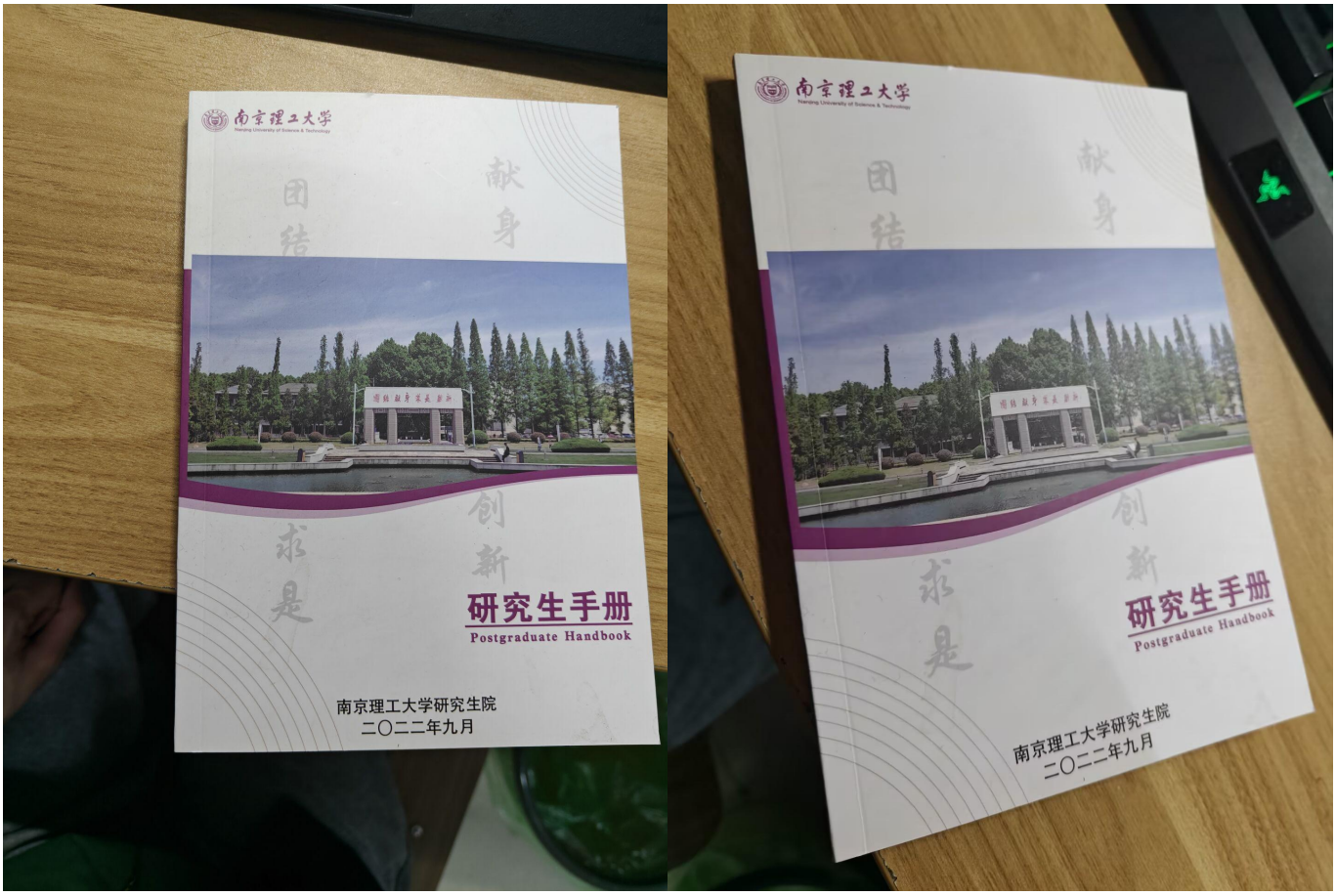
```
h, w, d = im1.shape
pts = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1, 0]]).reshape(-1, 1, 2)
dst = cv2.perspectiveTransform(pts, H)

img2 = cv2.polylines(im2, [np.int32(dst)], True, 255, 3, cv2.LINE_AA)
cv2.imshow("result", img2)
cv2.waitKey(0)
```

- 使用opencv中的单应性变换函数。

```
im1 = cv2.imread('left.jpg')
im2 = cv2.imread('right.jpg')

src_points = np.array([[581, 297], [1053, 173], [1041, 895], [558, 827]])
dst_points = np.array([[571, 257], [963, 333], [965, 801], [557, 827]])

H, _ = cv2.findHomography(src_points, dst_points)

h, w = im2.shape[:2]

im2_warp = cv2.warpPerspective(im2, H, (w, h))
```

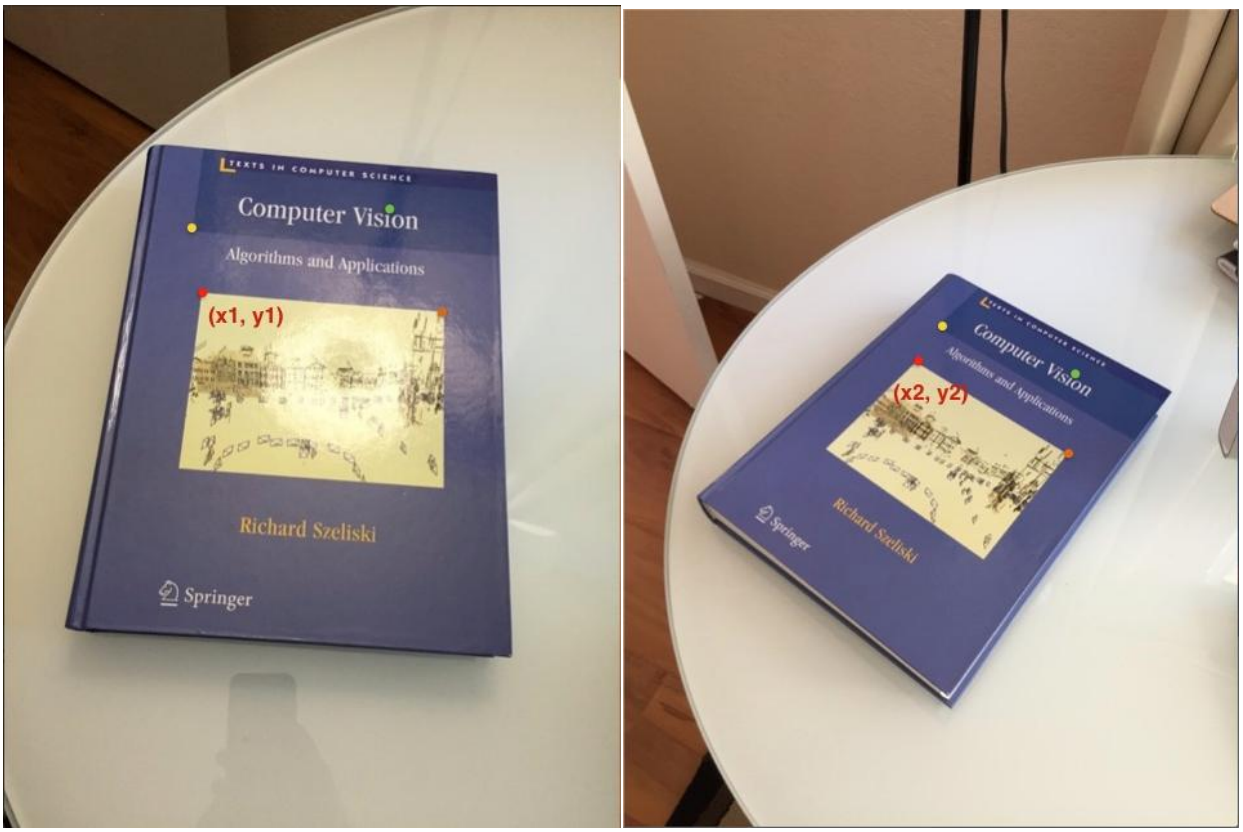
五. 实验结果

- 两张原图：

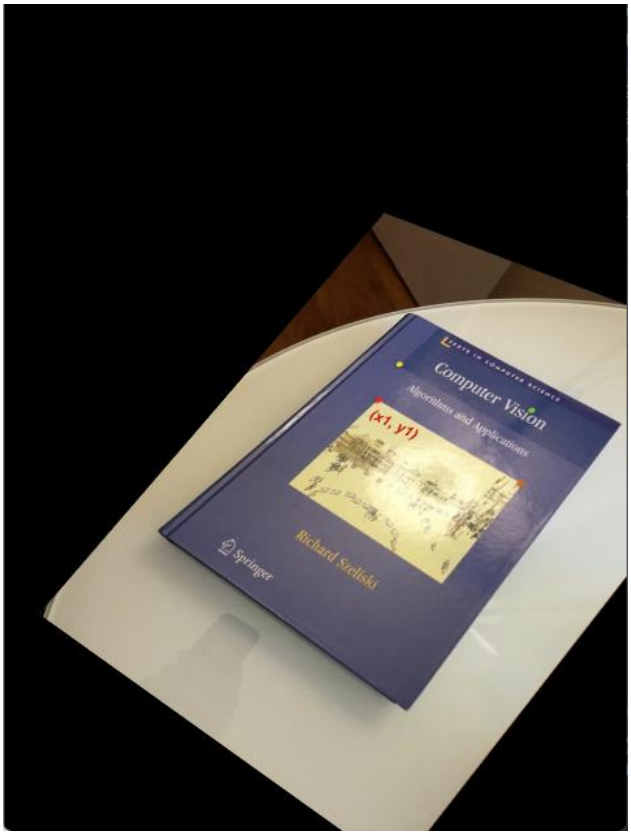


- 使用传统的寻找特征然后计算变换矩阵的做法得到结果：

- 使用opencv：



- 使用opencv实验结果：



六. 实验分析与总结

- 由实验结果可见，无论是使用特征点特征提取再选择比较好的匹配完成变换矩阵的计算还是通过opencv完成变换矩阵的计算以及变换，都能很好地将图片从一个视角转移到另一个视角上，完成两张图片的单应性变换。
- 在后续的实际应用中，可能需要根据具体的任务和场景选择适合的特征点提取算法、单应性计算算法以及变换方法，并对算法进行优化和改进，以获得更好的效果。就例如在特征点提取方面，可以使用深度学习算法来提高准确度。