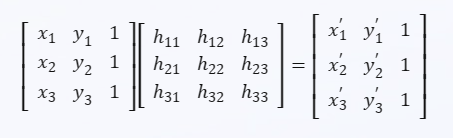
# 实验报告

## 实验目标

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。

## 实验过程

单应性变换是指在二维平面上的一种线性变换，它可以将一个平面上的点映射到另一个平面上的点。单应性变换可以用矩阵来表示，这个矩阵被称为单应性矩阵。



其中，H是一个3x3的矩阵，和是第i对匹配点的坐标。重点代码如下：

1. # 读取原图像
2. img\_src = cv2.imread('img\_src.png')
3. # 调用上述类，获取原图像的像素点
4. mouse1 = GetRoiMouse(img\_src)
5. mouse1.mouseclick()
6. # 为方便下一步运算，需要将像素点的类型转换为浮点型数据
7. pts\_src = np.float32(mouse1.lsPointsChoose)
8. # 读取目标图像
9. img\_dst = cv2.imread('img\_dst.png')
10. mouse2 = GetRoiMouse(img\_dst)
11. mouse2.mouseclick()
12. # 获取对应点
13. pts\_dst = np.float32(mouse2.lsPointsChoose)
14. # 目标图像的尺寸
15. dw, dh = img\_dst.shape[1], img\_dst.shape[0]
16. # 通过findHomography计算变换矩阵h
17. h, status = cv2.findHomography(pts\_src, pts\_dst, cv2.RANSAC, 5)
18. # 将变换矩阵h带入仿射变换实现矫正
19. img\_out = cv2.warpPerspective(img\_src, h, (dw, dh))

### 函数解析

#### 特征匹配

findHomography(srcPoints, dstPoints, method=None, ransacReprojThreshold=None, mask=None, maxIters=None, confidence=None)

srcPoints：源平面中点的坐标矩阵;

dstPoints：目标平面中点的坐标矩阵;

method：计算单应矩阵所使用的方法。不同的方法对应不同的参数，具体如下：

0： 利用所有点的常规方法；

RANSAC：基于RANSAC的鲁棒算法；

LMEDS：最小中值鲁棒算法；

RHO：基于PROSAC的鲁棒算法；

ransacReprojThreshold：将点对视为内点的最大允许重投影错误阈值（仅用于RANSAC和RHO方法）若srcPoints和dstPoints是以像素为单位的，则该参数通常设置在1到10的范围内。

mask：可选输出掩码矩阵，通常由鲁棒算法（RANSAC或LMEDS）设置。 请注意，输入掩码矩阵是不需要设置的；

maxIters：RANSAC算法的最大迭代次数，默认值为2000；

confidence：置信度，取值范围为0到1；

#### 透视变换

warpPerspective(src, M, dsize, dst=None, flags=None, borderMode=None, borderValue=None)

rc：输入图像矩阵

M：3\*3的透视变换矩阵，可以通过getPerspectiveTransform等函数获取

dsize：结果图像大小，为宽和高的二元组

dst：输出结果图像，可以省略，结果图像会作为函数处理结果输出

flags：可选参数，插值方法的组合（int 类型），默认值 INTER\_LINEAR

borderMode：可选参数，边界像素模式（int 类型），默认值 BORDER\_CONSTANT

borderValue：可选参数，边界填充值，当borderMode为cv2.BORDER\_CONSTANT时使用，默认值为None；

## 测试结果



左图为原图，中图为目标图，右图为变换图

从文字内容可以看出第三张图是由第一张变换而来的。其他部分和第二张图相似。