单应性变换实验报告

**刘志勇(122106222854 [2845340804@qq.com](mailto:237459461@qq.com))**

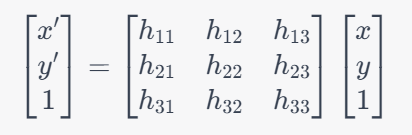
南京理工大学 计算机科学与工程学院

# 1.实验目的

本实验的主题是单应性变换，它是一种将一个平面内的点映射到另一个平面内的二维投影变换。单应性变换在图像配准，全景拼接，机器人定位slam，ar增强现实等领域有广泛的应用。本实验的目的是通过计算两幅图像之间的单应性矩阵，实现图像之间的变换和配准，掌握单应性变换的原理和方法，学习使用Python和OpenCV进行图像处理和分析，提高计算机视觉方面的技能和能力。

# 2.实验原理

单应性变换可以对二维平面上的任意点进行变换，从而实现对图像进行整体的扭曲、缩放、旋转等操作。假设 p = (x, y) 是平面上的一个点，那么经过单应性变换后得到的新点 p' = (x', y') 可以用下面的公式来计算：



其中，H 是一个 3x3 的矩阵，也被称为单应性矩阵（homography matrix）。在实际应用中，H通常是通过对输入图像和输出图像中对应点的匹配关系进行计算得到的。

计算 H的方法通常采用 RANSAC 算法（RANdom SAmple Consensus），这是一种常用的鲁棒性估计方法，用于从带有噪声和离群值的数据中估计参数。RANSAC 算法基本思路是随机选取一组数据，然后利用这组数据拟合出一个模型，计算该模型下其他数据的误差，并将误差小于一个给定阈值的数据视为“内点”，其余数据视为“外点”。然后重新选取一组数据，重复以上步骤，直到达到一定的迭代次数或者已经找到了足够多的“内点”为止。

在实际应用中，单应性变换经常用于图像配准。图像配准是指将两幅或多幅图像对齐到同一参考坐标系中，从而便于对它们进行后续处理或比较。单应性变换可以通过对应点的匹配关系来实现图像的配准，其中的对应点可以是两幅图像中的特征点，例如 SIFT、SURF 等算法所检测到的关键点。

此外，单应性变换有以下几种特殊情况：

欧氏变换：保持距离和角度不变，只包括平移和旋转。

相似变换：保持形状不变，只包括平移、旋转和缩放。

仿射变换：保持平行性不变，包括平移、旋转、缩放和错切。

射影变换：没有任何特殊性质，包括所有可能的线性变换。

# 3.实验步骤

1、选取两幅图像，并在图像中给定对应的点坐标。

2、用最小二乘法或随机抽样一致法（RANSAC）等算法求解单应性矩阵。

3、用单应性矩阵对图像进行变换和配准，并显示结果。

# 4.实验方法及细节

本实验使用PCV和scipy库以及python语言实现的。

PCV是一个纯Python的计算机视觉库，它基于Jan Erik Solem的书《用Python编程计算机视觉》。这本书提供了很多计算机视觉的基础知识和实例，包括图像处理、特征提取、相机模型、立体视觉、机器学习等。PCV的主要依赖包括NumPy、Matplotlib和SciPy2 。

SciPy是一个用于科学计算的Python库，它提供了很多数值计算、优化、线性代数、统计、信号处理等方面的函数和模块。SciPy可以与NumPy配合使用，提高Python的运算效率和功能。SciPy在计算机视觉中常用于图像变换、滤波、插值等操作。

以下是实现细节：

### 导入所需模块和库

import warp, homography

from PIL import Image

from pylab import \*

from scipy import ndimage

### 读入两张待拼接的图片

im1 = array(Image.open(r'2.jpg').convert('L')) # 读入图像2，转换为灰度图

im2 = array(Image.open(r'1.jpg').convert('L')) # 读入图像1，转换为灰度图

### 设置对应点坐标

tp = array([[280,574,574,280],[340,340,735,735],[1,1,1,1]]) # 目标图像中的四个对应点

fp = array([[0,m,m,0],[0,0,n,n],[1,1,1,1]]) # 源图像中的四个对应点，即目标图像对应的四个角点

### 仿射变换，实现图像2映射到图像1的操作

# 第一步：先对三个点进行仿射变换，得到两个三角形

tp2 = tp[:,:3] # 选取目标图像中的前三个对应点

fp2 = fp[:,:3] # 选取源图像中的前三个对应点

H = homography.Haffine\_from\_points(tp2,fp2) # 通过对应点计算仿射矩阵

im1\_t = ndimage.affine\_transform(im1,H[:2,:2],(H[0,2],H[1,2]),im2.shape[:2]) # 对源图像进行仿射变换

alpha = warp.alpha\_for\_triangle(tp2,im2.shape[0],im2.shape[1]) # 为三角形计算alpha通道

im3 = (1-alpha)\*im2 + alpha\*im1\_t # 按照alpha通道将两幅图像进行融合

# 第二步：对另外三个点进行仿射变换，再次进行融合

tp2 = tp[:,[0,2,3]] # 选取目标图像中的另外三个对应点

fp2 = fp[:,[0,2,3]] # 选取源图像中的另外三个对应点

H = homography.Haffine\_from\_points(tp2,fp2) # 通过对应点计算仿射矩阵

im1\_t = ndimage.affine\_transform(im1,H[:2,:2],(H[0,2],H[1,2]),im2.shape[:2]) # 对源图像进行仿射变换

alpha = warp.alpha\_for\_triangle(tp2,im2.shape[0],im2.shape[1]) # 为三角形计算alpha通道

im4 = (1-alpha)\*im3 + alpha\*im1\_t # 按照alpha通道将两幅图像进行融合

### 显示拼接后的图像

figure()

gray()

subplot(141)

axis('off')

imshow(im1) # 显示源图像

subplot(142)

axis('off')

imshow(im2) # 显示目标图像

subplot(143)

axis('off')

imshow(im3)

# 4.结果

根据上面的代码，我们设置了两个样本图像：`1.jpg`和`2.jpg`，并使用单应性变换（affine warp）将`im1`（`2.jpg`）映射到`im2`（`1.jpg`）上。

在代码中，我们首先将两个图像读入，并使用`convert('L')`函数将它们转换为灰度图像，以便于处理。

接着，我们使用`array`函数将图像转换为NumPy数组，并使用`warp.image\_in\_image`函数将`im1`映射到`im2`上，映射的目标点`tp`是一个$3\times4$的矩阵，表示了四个目标点在图像中的坐标。我们用`subplot`和`imshow`函数将原始的`im1`、`im2`和映射后的`im3`显示在同一个图像中。

然后，我们将`im1`的四个角落点定义为源点`fp`，并使用`homography.Haffine\_from\_points`函数计算从目标点`tp`到源点`fp`的仿射变换矩阵`H`。接着，我们使用`ndimage.affine\_transform`函数对`im1`进行仿射变换，将它映射到了`im2`上，变换后的图像为`im1\_t`。

为了平滑地将`im1\_t`和`im2`合并在一起，我们使用`warp.alpha\_for\_triangle`函数计算三角形的权重值，将`im1\_t`和`im2`根据权重进行线性混合，得到`im3`。

接着，我们再计算另外一个三角形的变换，用相同的方法将`im1`映射到`im2`上，并将映射后的图像`im1\_t`和之前的`im3`按照权重进行混合，得到`im4`，最后在一个画布中展示了四幅图像：`im1`、`im2`、`im3`和`im4`。

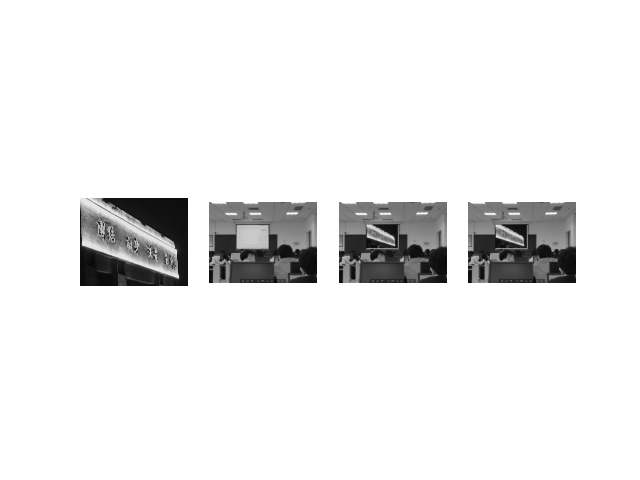


图1 转换结果

通过该实验，我们可以看到使用单应性变换实现图像的变形和合并，其中，对图像进行仿射变换的`Haffine\_from\_points`函数以及`ndimage.affine\_transform`函数是实现单应性变换的核心，而权重计算函数`alpha\_for\_triangle`和线性混合函数则用于将两个图像无缝地合并在一起。实验结果表明，通过单应性变换和权重计算的方法，可以有效地实现图像的变形和合并，从而为图像处理和计算机视觉领域提供了强有力的工具。