数字图像处理

Digital Image Processing

图像配准

自动化钱 61
柴嘉骏
2160405071

提交日期：2019 年 3 月 3 日

摘 要

本次作业为使用最小二乘法实现图像配准，具体操作为在两幅图像上找到对应的 n 对点，将其组成一个 3*n 的矩阵，通过最小二乘法推导出的公式获得仿射变换矩阵，最终实现图像配准。

本报告中和上次报告一样使用 Python3.5 下的 OpenCv 4.0 对图像进行处理。使用鼠标交互程序与其 *Callback* 函数获取两幅图像对应的特征点，在使用矩阵运算获取仿射变换矩阵并应用于原图像。

一．手动标点

使用如下程序获取鼠标点击图片的坐标并将其存储起来以便后续使用。

```
def on_EVENT_LBUTTONDOWN_A(event, x, y, flags, param):
    global P
    if event == cv2.EVENT_LBUTTONDOWN:
    xy = np.array((x, y, 1)).reshape([3, 1])
    P = np.concatenate((P, xy), axis=1)
    cv2.circle(imgA, (x, y), 1, (255, 0, 0), thickness=-1)
    cv2.putText(imgA, xy, (x, y), cv2.FONT_HERSHEY_PLAIN,
                1.0, (0, 0, 0), thickness=1)
    cv2.imshow("imageA", imgA)
```

二．输出两幅图中对应点的坐标

得到的数据如下所示，其中 P 为待变换图像对应的矩阵，Q 为目标图像对应的矩阵：

```
P: [[1022. 1021. 1023. 1210. 1209. 1209. 1194.]
  [1768. 1810. 2126. 2241. 1879. 1830. 1694.]
  [   1.    1.    1.    1.    1.    1.    1.]]
Q: [[ 717.  705.  628.  777.  871.  885.  904.]
   [1279. 1319. 1624. 1785. 1433. 1388. 1251.]
   [   1.    1.    1.    1.    1.    1.    1.]]
```

三．计算转换矩阵

使用 ndarray 的矩阵运算方法计算仿射变换矩阵

```
H1 = Q.dot(P.transpose())
H2 = np.linalg.inv(P.dot(P.transpose()))
H = H1.dot(H2)[:2]
```

得到仿射变换矩阵如下所示：

```
H: [[    0.97232037   -0.25818311  180.50929908]
   [    0.25938206    0.96658498 -695.56885885]]
```

四．输出转换之后的图像

五．代码示例：

```
# coding: utf-8
import cv2
import numpy as np
imgA = cv2.imread('/home/hero/Documents/DIP-Homework/Homework1-2.22/Requirement/ImageA.jpg')
imgB = cv2.imread('/home/hero/Documents/DIP-Homework/Homework1-2.22/Requirement/ImageB.jpg')
height, width = imgA.shape[0], imgA.shape[1]
P = np.empty(shape=[3, 0])
Q = np.empty(shape=[3, 0])
def on_EVENT_LBUTTONDOWN_A(event, x, y, flags, param):
    global P
    if event == cv2.EVENT_LBUTTONDOWN:
    xy = np.array((x, y, 1)).reshape([3, 1])
    P = np.concatenate((P, xy), axis=1) cv2.circle(imgA, (x, y), 1, (255, 0, 0), thickness=-1)
    cv2.putText(imgA, xy, (x, y), cv2.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 0), thickness=1)
    cv2.imshow("imageA", imgA)
def on_EVENT_LBUTTONDOWN_B(event, x, y, flags, param):
    global Q
    if event == cv2.EVENT_LBUTTONDOWN:
    xy = np.array((x, y, 1)).reshape([3, 1])
    Q = np.concatenate((Q, xy), axis=1)
    cv2.circle(imgB, (x, y), 1, (255, 0, 0), thickness=-1)    cv2.putText(imgB, xy, (x, y),
cv2.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 0), thickness=1)
```

```
    cv2.imshow("imageB", imgB)
cv2.namedWindow("imageA", 0)
cv2.resizeWindow("imageA", 640, 480)
cv2.setMouseCallback("imageA", on_EVENT_LBUTTONDOWN_A)
cv2.imshow("imageA", imgA)
cv2.waitKey(0)
cv2.destroyAllWindows()

cv2.namedWindow("imageB", 0)
cv2.resizeWindow("imageB", 640, 480)
cv2.setMouseCallback("imageB", on_EVENT_LBUTTONDOWN_B) cv2.imshow("imageB", imgB)
cv2.waitKey(0)
cv2.destroyAllWindows()

H1 = Q.dot(P.transpose())
H2 = np.linalg.inv(P.dot(P.transpose()))
H = H1.dot(H2)[:2] np.set_printoptions(suppress=True)

print("P:", P)
print("Q:", Q)
print("H:", H)

registrated_img = cv2.warpAffine(imgA, H, (height, width))
cv2.imwrite('/home/hero/Documents/DIP-Homework/Homework1-2.22/Requirement/Registration.bmp',
registrated_img)
```

六．心得体会

本次实验使用 Python 和 OpenCv 有点难进行，因为 Python 对于矩阵操作不是十分友善，而且在鼠标坐标的获取上我也不是很清楚，因此在获取坐标点上耽误了一些时间。

但总体来说，这次实验的收获还是很大的，只要是做自己之前不会的东西，能多了解到一些知识就足够了。