#### 计算机视觉实践-练习1

学号：122106222783 姓名：王心怡

1 实验目标

·理解关键点检测算法DOG原理。

·理解尺度变化不变特征SIFT。

·采集一系列局部图像，自行设计拼接算法。

·使用Python实现图像拼接算法。

2 实现说明

**·从SIFT中检测特征和关键点**

def Detect\_Feature\_And\_KeyPoints(self, image):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# 从图像中检测和提取特征

descriptors = cv2.SIFT\_create()

(Keypoints, features) = descriptors.detectAndCompute(image, None)

Keypoints = np.float32([i.pt for i in Keypoints])

return (Keypoints, features)

**·得到有效匹配点**

def matchKeypoints(self, KeypointsA, KeypointsB, featuresA, featuresB,lowe\_ratio, max\_Threshold):

AllMatches = self.get\_Allpossible\_Match(featuresA,featuresB);

valid\_matches = self.All\_validmatches(AllMatches,lowe\_ratio)

if len(valid\_matches) > 4:

# 构造两组点

pointsA = np.float32([KeypointsA[i] for (\_,i) in valid\_matches])

pointsB = np.float32([KeypointsB[i] for (i,\_) in valid\_matches])

(Homograpgy, status) = self.Compute\_Homography(pointsA, pointsB, max\_Threshold)

return (valid\_matches, Homograpgy, status)

else:

return None

def get\_Allpossible\_Match(self,featuresA,featuresB):

# 使用欧氏距离和opencv计算所有匹配

match\_instance = cv2.DescriptorMatcher\_create("BruteForce")

All\_Matches = match\_instance.knnMatch(featuresA, featuresB, 2)

return All\_Matches

def All\_validmatches(self,AllMatches,lowe\_ratio):

# 根据lowe概念获取所有有效匹配

valid\_matches = []

for val in AllMatches:

if len(val) == 2 and val[0].distance < val[1].distance \* lowe\_ratio:

valid\_matches.append((val[0].trainIdx, val[0].queryIdx))

return valid\_matches

def Compute\_Homography(self,pointsA,pointsB,max\_Threshold):

# 使用两幅图像中的点计算单应性

(H, status) = cv2.findHomography(pointsA, pointsB, cv2.RANSAC, max\_Threshold)

return (H,status)

**·计算单应性获得图像的透视图**

def getwarp\_perspective(self,imageA,imageB,Homography):

val = imageA.shape[1] + imageB.shape[1]

result\_image = cv2.warpPerspective(imageA, Homography, (val , imageA.shape[0]))

return result\_image

**·可视化关键匹配点**

def draw\_Matches(self, imageA, imageB, KeypointsA, KeypointsB, matches, status):

(hA,wA) = self.get\_image\_dimension(imageA)

vis = self.get\_points(imageA,imageB)

# 在匹配点上循环

for ((trainIdx, queryIdx), s) in zip(matches, status):

if s == 1:

ptA = (int(KeypointsA[queryIdx][0]), int(KeypointsA[queryIdx][1]))

ptB = (int(KeypointsB[trainIdx][0]) + wA, int(KeypointsB[trainIdx][1]))

cv2.line(vis, ptA, ptB, (0, 255, 0), 1)

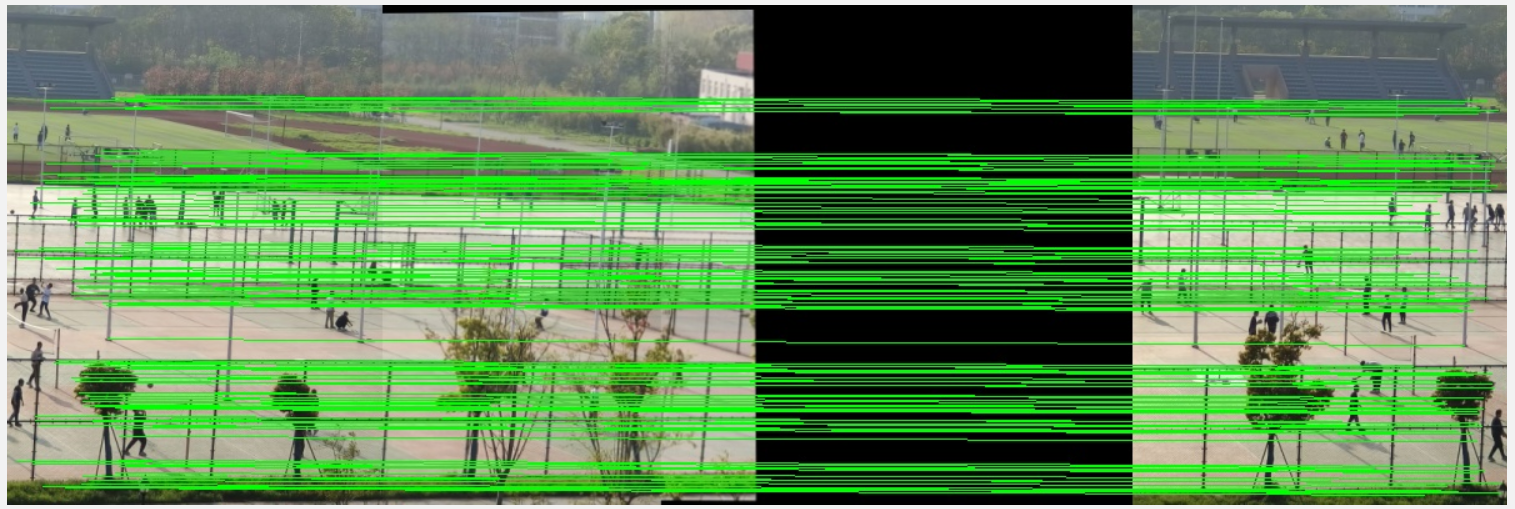
return vis

3 结果截图

原始图像：

匹配点：

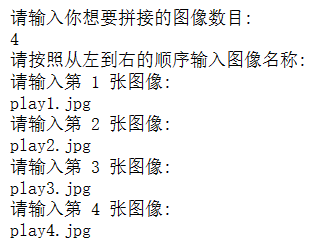


拼接结果：



4 运行说明

·将需要拼接的图像放在main.ipynb所在文件下，运行main.ipynb，根据提示输入图像数目和图像名称。



·待程序运行完后，文件夹里会出现2个运行结果，分别为匹配点Matched\_points.jpg和拼接得到的全景图像Panorama\_image.jpg。

