《计算机视觉》实验报告

姓名：汪江豪 学号：22121630

实验七

一. 任务 1

1、完成 SIFT 特征匹配，实现图片拼接，包括以下步骤

（1）输入两张同一场景不同视角拍摄的图片

（2）分别提取图片的 SIFT 特征

（3）关键点匹配

（4）采用 RANSAC 算法进行提纯

（5）获取两张图片的变换关系（单应性矩阵，可通过 4 对匹配点求出），完成拼接（选做加分）

以上步骤都需要给出实验结果图

a) 核心代码：

1. 创建 SIFT 对象，检测并绘制关键点

```
# 创建 SIFT 对象
sift = cv2.SIFT_create()
# 检测关键点
kp1, des1 = sift.detectAndCompute(gray1, None)
kp2, des2 = sift.detectAndCompute(gray2, None)
# 绘制关键点
img1_kp = cv2.drawKeypoints(img1, kp1, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
img2_kp = cv2.drawKeypoints(img2, kp1, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

2. 使用 FLANN 匹配器进行特征匹配

```
# 使用 FLANN 匹配器进行特征匹配
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
```

```
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)
good_matches = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        good_matches.append(m)
```

3. 使用 RANSAC 算法进行匹配点的提纯

```
# 提取匹配点的坐标
src_pts = np.float32([kp1[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
dst_pts = np.float32([kp2[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)
# 使用 RANSAC 算法进行匹配点的提纯
M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
#根据 RANSAC 筛选结果，绘制匹配点连线图，只显示内点匹配
matchesMask = mask.ravel().tolist()
draw_params = dict(matchColor=(0, 255, 0),singlePointColor=None,
matchesMask=matchesMask,flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
img_ransac_matches = cv2.drawMatches(img1, kp1, img2, kp2, good_matches,
None, **draw_params)
```

4. 完成图片拼接

```
# 根据单应性矩阵完成图片拼接
h1, w1 = img1.shape[:2]
h2, w2 = img2.shape[:2]
# 计算拼接后图像的大小
points_img1 = np.float32([[0, 0], [0, h1], [w1, h1], [w1, 0]]).reshape(-1, 1, 2)
points_img2 = np.float32([[0, 0], [0, h2], [w2, h2], [w2, 0]]).reshape(-1, 1, 2)
points_img2_transformed = cv2.perspectiveTransform(points_img2, M)
points_combined = np.concatenate((points_img1, points_img2_transformed), axis=0)

[x_min, y_min] = np.int32(points_combined.min(axis=0).ravel() - 0.5)
[x_max, y_max] = np.int32(points_combined.max(axis=0).ravel() + 0.5)

translation_dist = [-x_min, -y_min]
H_translation = np.array([[1, 0, translation_dist[0]], [0, 1, translation_dist[1]], [0, 0, 1]])

# 将 img1 和 img2 拼接到同一图像中
result = cv2.warpPerspective(img1, H_translation @ M, (x_max - x_min, y_max - y_min))
```

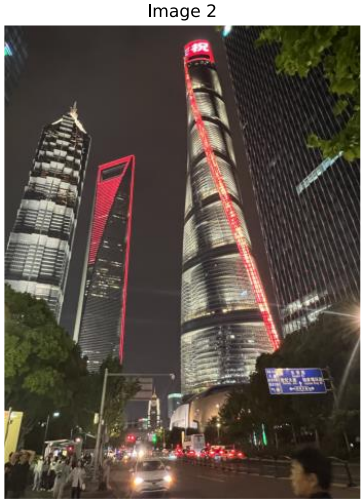
```
# 确保目标区域大小与 img2 一致
y1, y2 = translation_dist[1], translation_dist[1] + h2
x1, x2 = translation_dist[0], translation_dist[0] + w2
# 调整切片范围，确保不超出 result 的边界
y2 = min(y2, result.shape[0])
x2 = min(x2, result.shape[1])
result[y1:y2, x1:x2] = img2[:y2 - y1, :x2 - x1]
```

b) 实验结果截图

1. 两张同一场景不同视角拍摄的图片：

Image 1

Image 2



2. 提取图片的 SIFT 特征后：

Image 1 keypoints

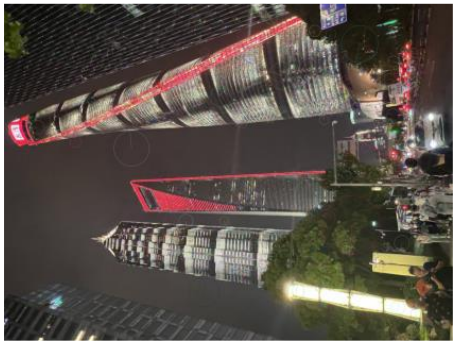
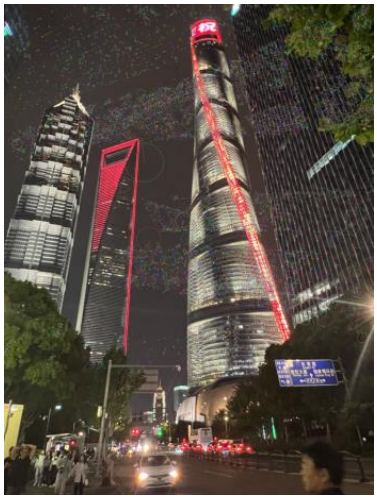
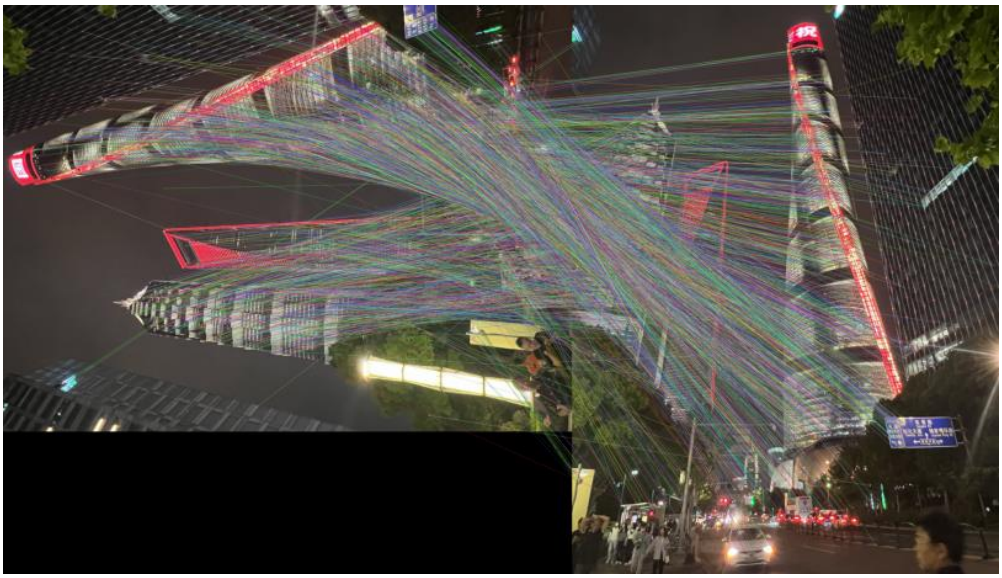


Image 2 keypoints



3. 关键点匹配：

Feature Matches

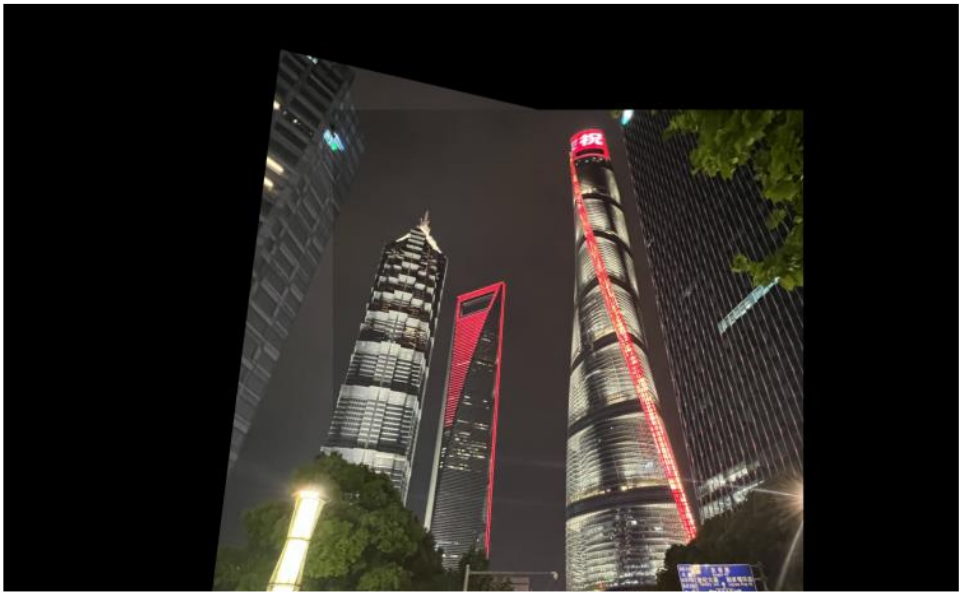


4. 采用 RANSAC 算法进行提纯：

RANSAC Matches

5. 图片拼接结果：

Stitched Image



c) 实验小结

在计算机视觉领域中，图像拼接式一项基础且重要的技术，广泛应用于全景图像生成、医学影响分析、遥感图像处理等领域。本次实验旨在通过 SIFT 特征匹配，并经过 RANSAC 算法进行提纯，实现了两幅图像较好的拼接效果。但如果两幅图像角度差异过大，拼接效果将很不理想，例如，光照不均、图像变形、噪声干扰等可能影响特征提取和匹配的精度，需要进一步研究和解决。