《计算机视觉》实验报告

姓名：冯俊佳 学号：23122721

实验 7 SIFT 特征

一. 任务 1

a) 核心代码：

```
# Step1 图像加载与预处理
def load_image(img1_path, img2_path):
    img1 = cv.imread(img1_path)
    img2 = cv.imread(img2_path)
    # 统一分辨率
    rate = 600 / img1.shape[1]
    img1 = cv.resize(img1, (int(rate*img1.shape[1]), int(rate*img1.shape[0])))
    img2 = cv.resize(img2, (img1.shape[1], img1.shape[0]))
    # 边界填充（用于拼接时留出空间）
    img1 = cv.copyMakeBorder(img1, 250,250,250,250, cv.BORDER_CONSTANT)
    img2 = cv.copyMakeBorder(img2, 250,250,250,250, cv.BORDER_CONSTANT)
    return img1, img2

# Step2  特征提取与匹配
def match_feature_point(img1, img2):
    # SIFT 特征检测
    sift = cv.SIFT_create()
    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)

    # FLANN 匹配器
    flann = cv.FlannBasedMatcher(
        dict(algorithm=1, trees=5),  # KD-Tree 索引
        dict(checks=50)  # 搜索次数
    )
    matches = flann.knnMatch(des1, des2, k=2)  # KNN 匹配
    return kp1, kp2, matches

```

```
# Step3 匹配点提纯与单应性矩阵计算
def get_good_match(img1, img2, kp1, kp2, matches):
    # 比率测试筛选匹配点
    good_match = []
    for m, n in matches:
        if m.distance < 0.5 * n.distance:  # Lowe's ratio test
            good_match.append(m)

    # 计算单应性矩阵（RANSAC 提纯）
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good_match]).reshape(-1, 1, 2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good_match]).reshape(-1, 1, 2)
    M, _ = cv.findHomography(src_pts, dst_pts, cv.RANSAC, 5.0)

    # 图像配准
    img2_warped = cv.warpPerspective(img2, M, (img2.shape[1], img2.shape[0]))

    # 简单拼接（直接覆盖）
    dst = img1.copy()
    dst[img2_warped > 0] = img2_warped[img2_warped > 0]

    return M, dst

# Step4 图像融合（加权混合）
def blend_image(img1, img2):
    rows, cols = img1.shape[:2]
    result = np.zeros((rows, cols, 3), np.uint8)

    # 找到重叠区域边界
    left = next(col for col in range(cols) if img1[:, col].any() and img2[:, col].any())
    right = next(col for col in reversed(range(cols)) if img1[:, col].any() and img2[:, col].any())

    # 线性加权融合
    for col in range(cols):
        if img1[:, col].any() and img2[:, col].any():
            alpha = (col - left) / (right - left)  # 动态权重
            result[:, col] = img1[:, col] * alpha + img2[:, col] * (1 - alpha)
```

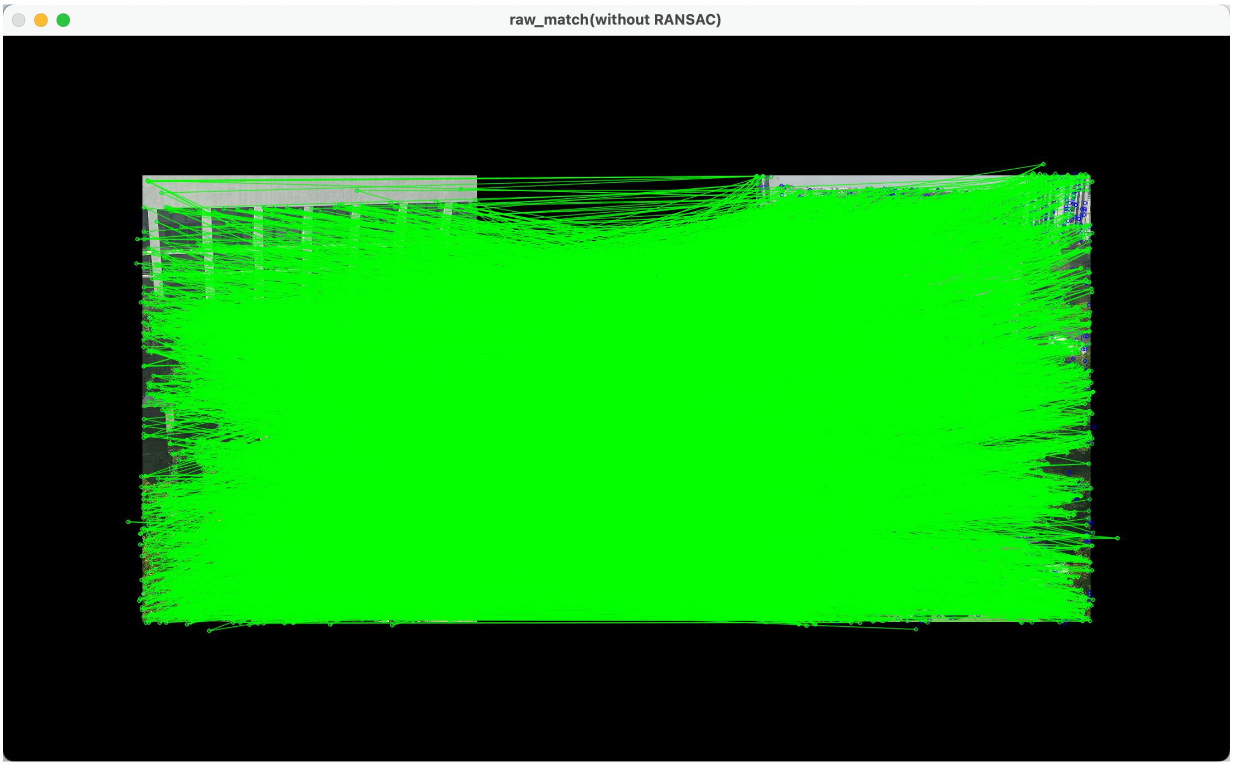
```
        elif img1[:, col].any():
            result[:, col] = img1[:, col]
        else:
            result[:, col] = img2[:, col]
    return result
```

b) 实验结果截图

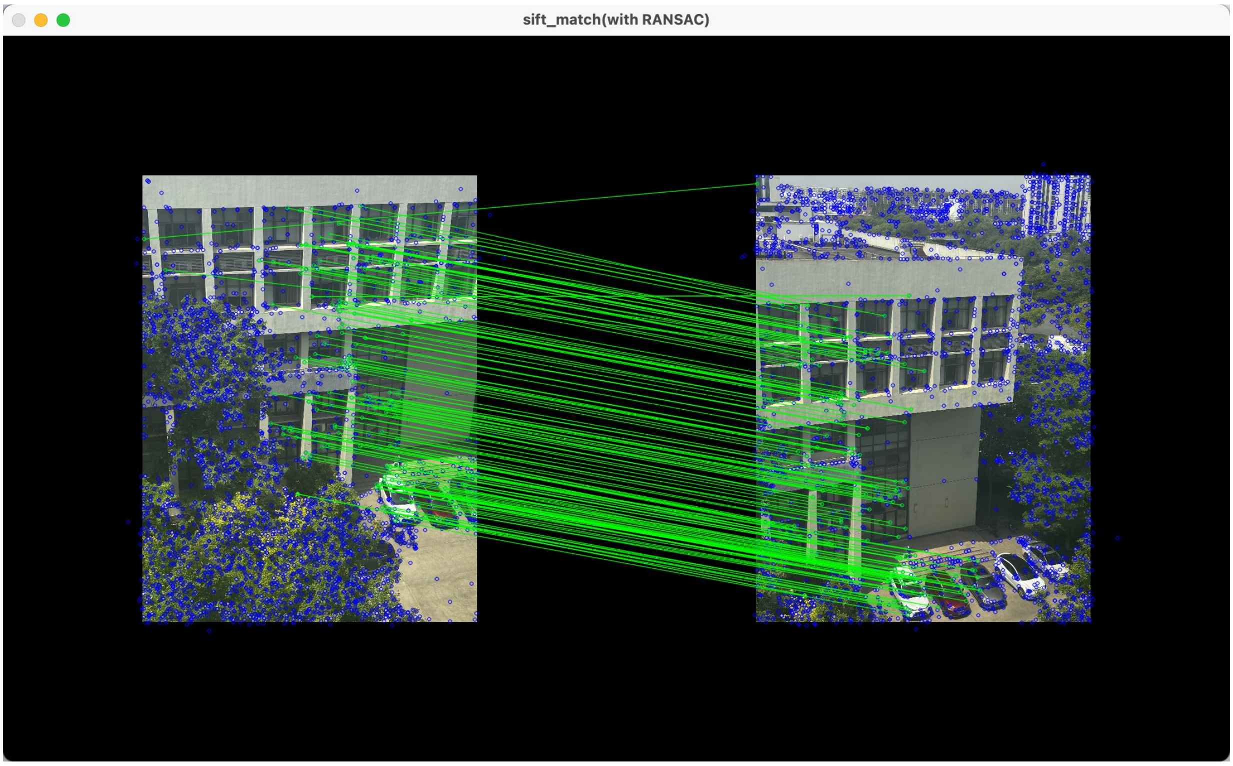
Step1 输入两张同一场景不同视角拍摄的图片

Step2 分别提取图片的 SIFT 特征

Step3 关键点匹配（未用 RANSAC 提纯版）

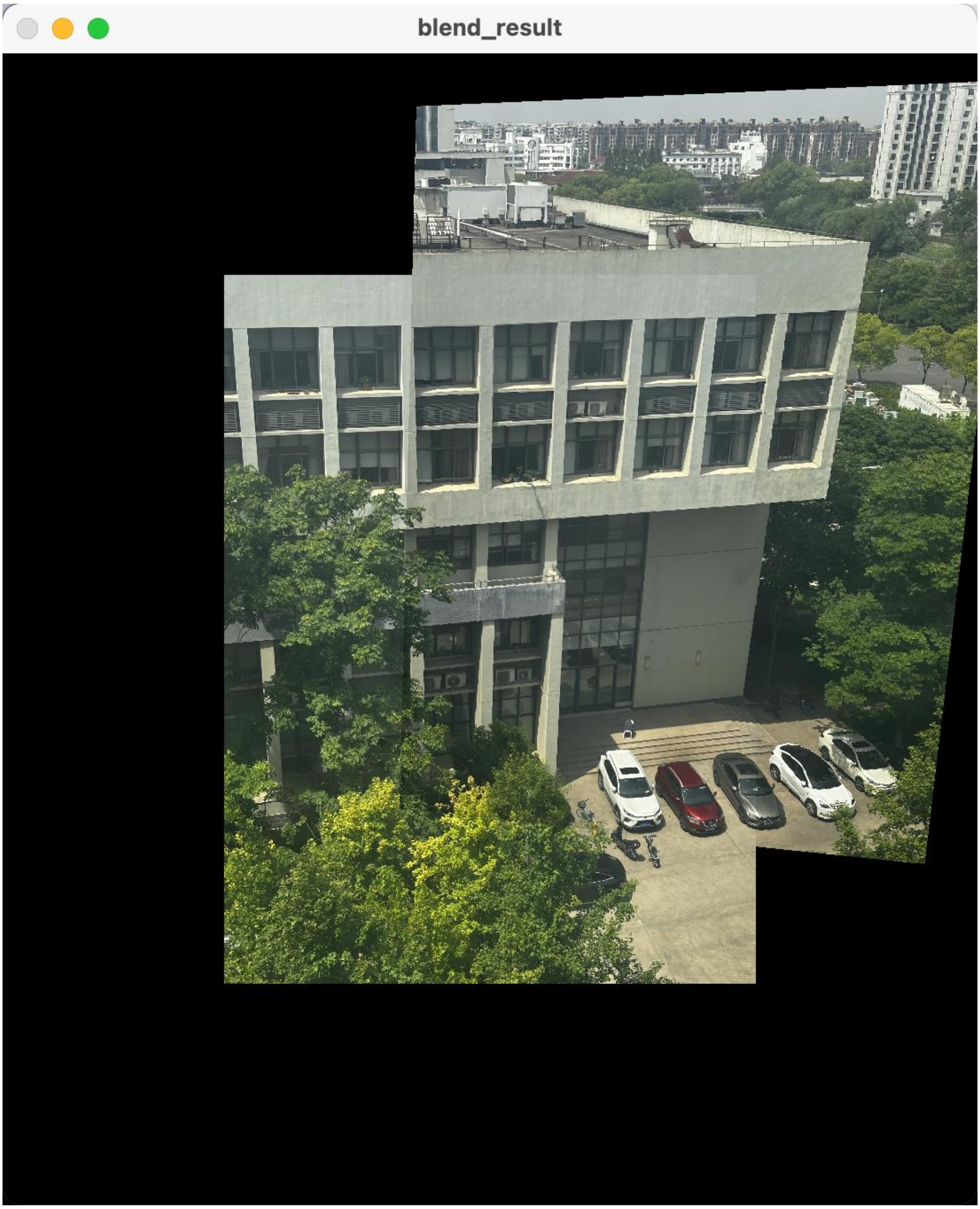


Step4 采用 RANSAC 算法进行提纯



Step5 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

```
单应性矩阵:
 [[ 1.12790993e+00 -1.07955738e-02 -2.55370787e+02]
 [ 9.11004872e-02  9.34234513e-01  1.72211730e+02]
 [ 1.94995855e-04 -1.46831127e-04  1.00000000e+00]]
```



c) 实验小结

通过本实验，我深入理解了基于 SIFT 特征的图像配准原理，掌握了 RANSAC 和单应性变换在计算机视觉中的重要应用。在实验过程中，我也遇到了一些困难，比如匹配点不足，发现是由于部分图像因纹理稀疏导致匹配对数少于 10。对此，我调整了图像分辨率（实验中发现 600px 宽度效果最佳），并通过修改 contrastThreshold 参数来增加 SIFT 特征点数量。