1. 简介

在计算机视觉领域，**运动结构恢复**（Structure from Motion, SfM）是一种从二维图像序列中重建三维场景结构和估计相机运动的方法。**束调整**（Bundle Adjustment, BA）则用于优化相机参数和三维点的位置，以最小化投影误差。在本次实验中，我们将实现 SfM 算法并使用束调整来提升重建的精度。

2. 数据预处理

在进行 SfM 之前，必须进行数据预处理步骤，包括关键点检测、特征匹配和几何验证。

2.1 关键点检测

我们使用了 SIFT（尺度不变特征变换）算法来检测图像中的关键点。SIFT 算法能够提取图像中的独特特征，并生成描述符用于后续的特征匹配。具体实现如下：

```
image = cv2.imread(image_file)
sift = cv2.SIFT_create()
keypoints, descriptors = sift.detectAndCompute(image, None)
```

2.2 特征匹配

检测到关键点后，我们通过暴力匹配器（BFMatcher）进行特征匹配，并应用 Lowe 比率测试来过滤匹配结果。Lowe 比率测试通过比较最佳匹配和次佳匹配的距离比率来确定匹配的有效性：

```
bf = cv2.BFMatcher()
matches = bf.knnMatch(descriptors1, descriptors2, 2)
good_matches = [m for m, n in matches if m.distance < lowe_ratio * n.distance]
```

2.3 几何验证

使用 RANSAC（随机抽样一致性）方法对匹配结果进行几何验证，以消除错误匹配。RANSAC 通过估计本质矩阵来滤除不符合几何约束的匹配：

```
essential_mtx,is_inlier=cv2.findEssentialMat(points1,points2, cameraMatrix=camera_intrinsics, method=cv2.RANSAC, threshold=ransac_threshold)
```

3. 场景图构建

在构建场景图时，我们将图像视为节点，匹配对视为边，只有在两幅图像之间存在足够的内点匹配时才添加边。场景图有助于选择最佳的初始图像对以进行增量式 SfM。

4. 运动结构恢复（Structure from Motion）

4.1 初始化

初始化阶段，我们选择拥有最多内点匹配的图像对作为初始图像对。使用本质矩阵恢复相机的相对姿态（旋转和平移）：

```
_, R, t, _ = cv2.recoverPose(E=essential_mtx, points1=points2d_1, points2=points2d_2, cameraMatrix=intrinsics)
extrinsics2 = np.concatenate((R, t), axis=1)
```

4.2 增量 SfM

在增量 SfM 中，我们逐步注册新的图像，恢复相机姿态并通过三角化计算新图像的 3D 点位置。每次迭代，我们选择与当前已注册图像中匹配数量最多的未注册图像进行注册。

```
new_points3d = triangulate(image_id1, image_id2, kp_idxs1, kp_idxs2, extrinsics1, extrinsics2, intrinsics)
```

4.3 三角化

三角化是根据已知的相机位姿和匹配点，在空间中计算 3D 点的位置。我们使用 OpenCV 的 triangulatePoints 函数实现该步骤：

```
points3d = cv2.triangulatePoints(projMatr1=proj_mtx1, projMatr2=proj_mtx2,
```

projPoints1=proj_pts1.T, projPoints2=proj_pts2.T)
points3d = points3d[:3] / points3d[3]

5. 束调整（Bundle Adjustment）

束调整用于在增量 SfM 完成后进一步优化相机参数和 3D 点的位置，目的是最小化所有图像中 2D 点与其对应 3D 点投影之间的重投影误差。

5.1 实现细节

我们在 compute_ba_residuals 函数中计算重投影误差：

calculated_points2d = np.einsum('ijk,ki->ij', P, homo_3d_points_T)
calculated_points2d /= calculated_points2d[:, -1].reshape((calculated_points2d.shape[0], 1))
residuals = np.linalg.norm(points2d - calculated_points2d, axis=1)

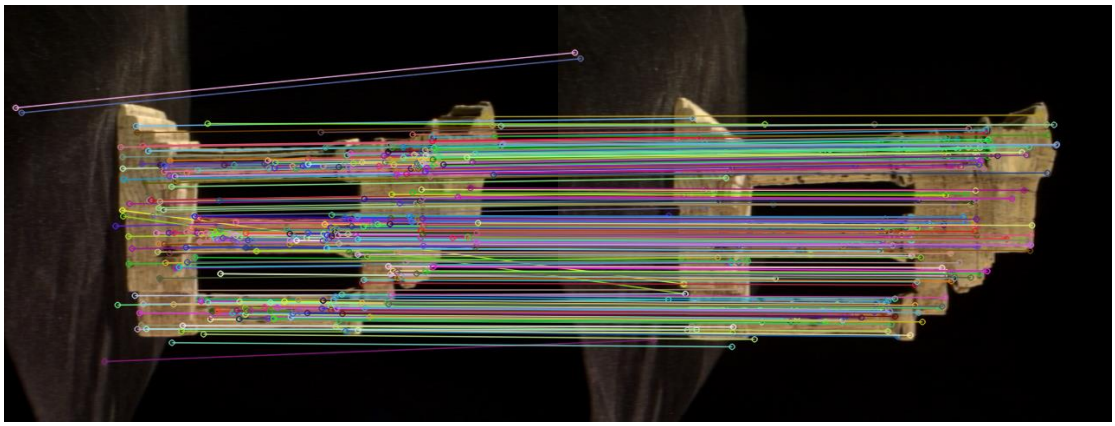
5.2 优化过程

优化通过最小化总残差来更新相机姿态和 3D 点坐标：

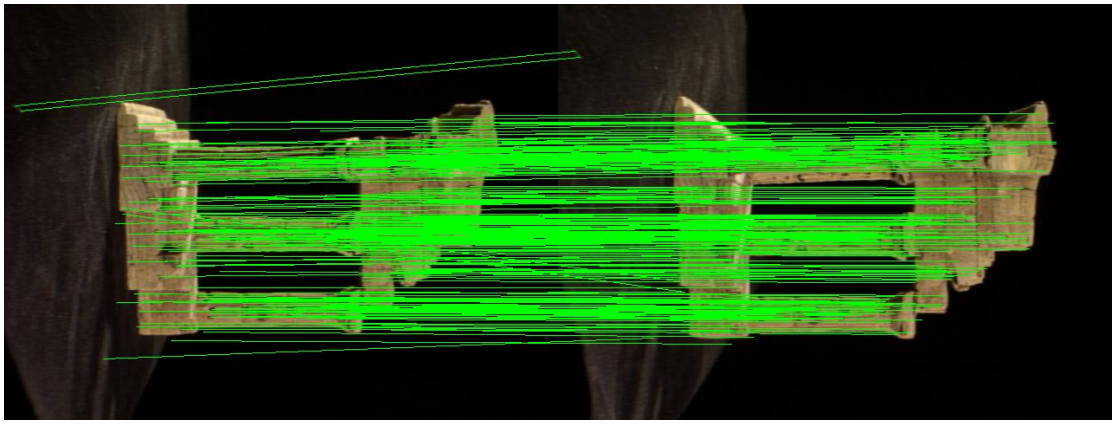
results=least_squares(fun=compute_ba_residuals,x0=parameters,method='trf',
max_nfev=max_nfev, args=(intrinsics, num_cameras, points2d, camera_idxs, points3d_idxs))

6. 实验结果

6.1 特征匹配的图像



6.2 RANSAC 过滤后的图像



6.3 束调整后的相机路径变化

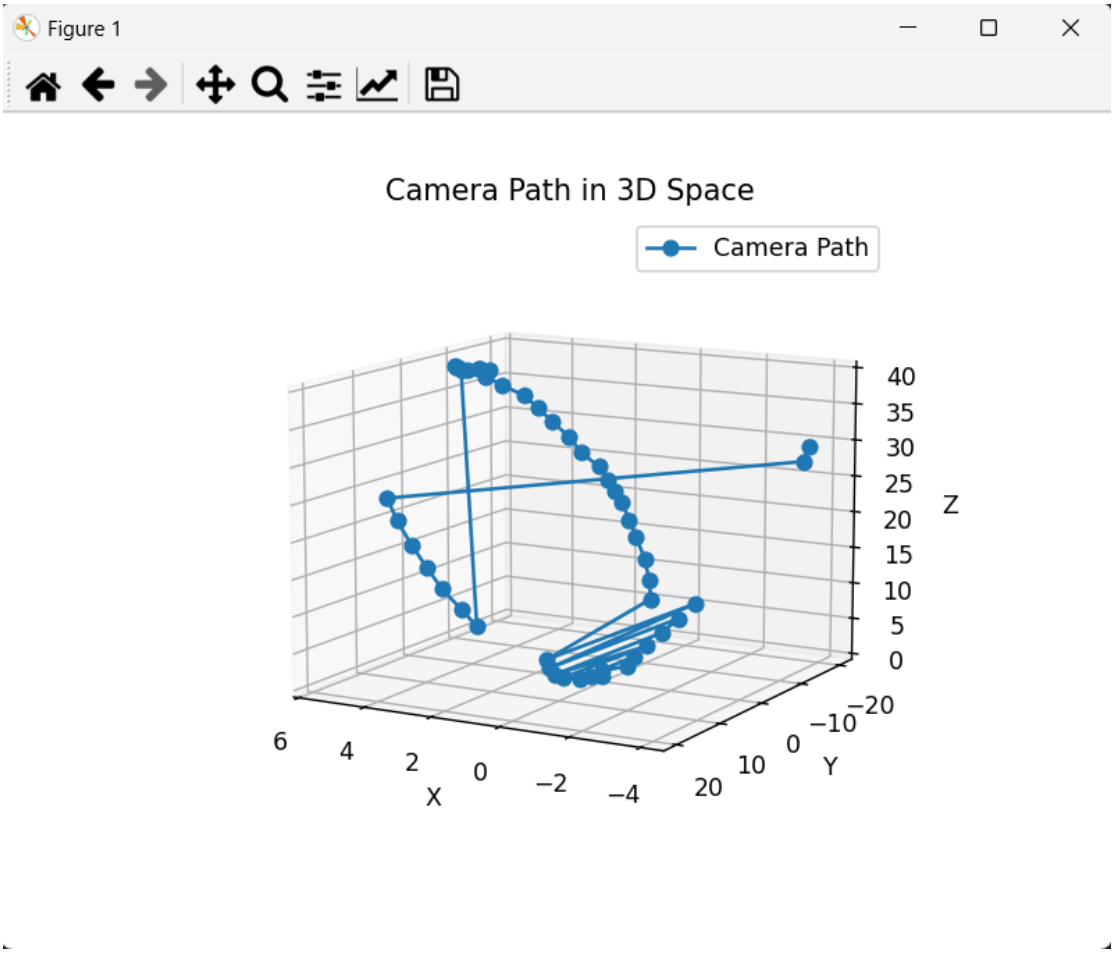
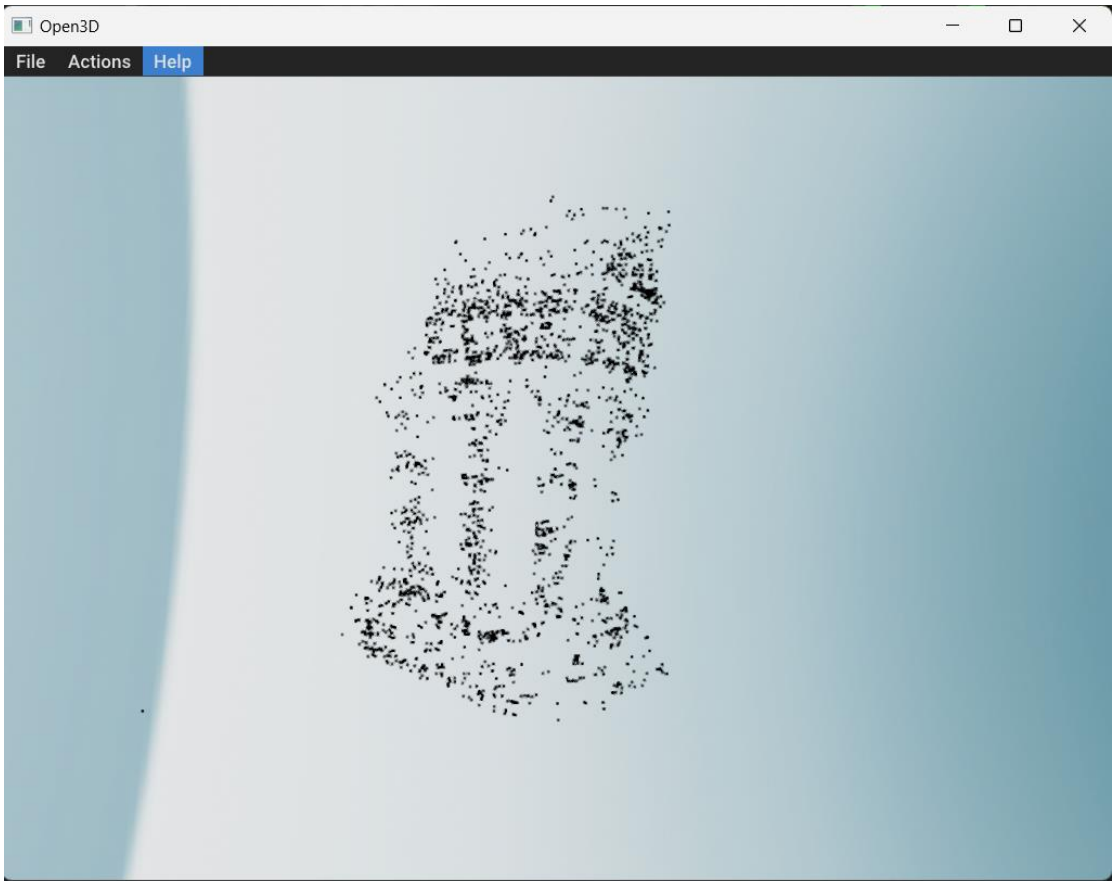


Figure 1
Camera Path in 3D Space
Camera Path
40
35
30
25
20
15
10
5
0
Z
6
4
2
0
−2
−4
X
20
10
0
−10
−20
Y

6.4 重建的 3D 点云


Open3D
File
Actions
Help