
摘要

在本次三维重建作业中，我们采用了一套完整的基于传统计算机视觉的流程，实现了对简单数据的精确三维场景重建。

首先，我们利用尺度不变特征变换（SIFT）算法对图像进行特征提取，通过构建高斯金字塔和差分金字塔来识别和定位关键特征点，并生成对应的 SIFT 描述子。接着，通过 FLANN 匹配器结合 Lowe 比率测试，筛选出稳定且准确的匹配对。

在特征匹配的基础上，我们采用了对极几何方法进行场景初始化，计算基础矩阵和本质矩阵，恢复相机姿态，并通过三角测量法重建出初始的三维点坐标。此外，我们还使用 PnP 算法进一步对场景进行重建，通过求解相机的外参矩阵和应用三角定位法，优化了三维点的坐标。

为了提升重建结果的精度，我们应用了光束平差法（Bundle Adjustment）对场景进行优化。我们构建了投影模型，定义了误差函数，并采用非线性最小二乘方法对三维点坐标和相机参数进行联合优化，有效减少了重投影误差。

在实验探究中，我们深入分析了特征提取和匹配中的参数选择对重建效果的影响，以及不同算法选择对特征匹配和场景重建的影响。我们还探讨了去除 Bundle Adjustment 模块和仅使用对极几何方法进行三维重建的局限性，指出了误差累积和尺度因子对齐在重建过程中的重要性。

通过本项工作，我们不仅熟悉了三维重建技术，还对如何优化和改进重建流程有了更为深刻的理解。我们的实验结果表明，综合运用 SIFT 特征提取、特征匹配、对极几何和 Bundle Adjustment 等技术，可以显著提高三维重建的质量和精度。

目录

1 引言	3
2 方法	3
2.1 超参数说明与设置	3
2.2 SIFT: Features 提取	5
2.2.1 理论分析	5
2.2.2 实际实现	5
2.3 图像特征匹配	6
2.3.1 理论分析	6
2.3.2 实际实现	7
2.4 对极几何场景初始化	8
2.4.1 理论分析	8
2.4.2 实际实现	8
2.5 使用 Perspective-n-Point 进行场景重建	9
2.5.1 理论分析	9

2.5.2	实际实现	10
2.6	Bundle Adjustment 进行场景优化	10
2.6.1	理论分析	10
2.6.2	实际实现	11
3	结果	12
3.1	特征提取展示	12
3.2	特征匹配展示	12
3.3	场景初始化	12
3.4	PnP 结果展示	13
3.5	Bundle Adjustment 结果展示	14
4	讨论	14
4.1	不使用 Bundle Adjustment	14
4.2	对特征点提取中阈值参数选取的分析	15
4.3	对特征点匹配中阈值参数选取的分析	18
4.4	对 <code>cv2.findFundamentalMat(p1, p2, alg)</code> 中算法选择的探究	18
4.5	仅使用对极几何进行三维重建	18
5	总结	21

1 引言

在本次作业中，我们初步探索了基于传统计算机视觉技术的三维重建，旨在通过对图像数据的精确分析，实现对现实世界场景的三维模型构建。我们首先采用了基于尺度不变特征变换（SIFT）的图像特征提取技术，结合 FLANN 匹配器和 Lowe 比率测试，以确保特征点匹配的准确性和稳定性。进一步地，通过对极几何方法和 PnP 算法的应用，初始化了场景并优化了相机姿态，为三维点的精确重建奠定了基础。最终，通过光束平差法（Bundle Adjustment）的细致优化，进一步显著提高了重建结果的精度。此外，我们还进一步进行了消融实验，比较了不同模块的作用，并对参数选择和算法优化进行了一定的探讨¹。

2 方法

2.1 超参数说明与设置

¹代码：<https://github.com/huskydoge/AI4701-ComputerVision-3D-Recon>

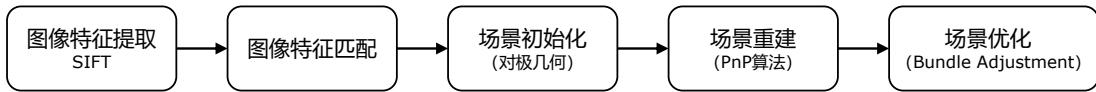


图 1: 整体流程图

```

1 GLOB:
2     intrinsic_path: camera_intrinsic.txt # 相机内参矩阵路径
3     world_cam_path: images/0000.png # 设置0000号相机作为世界坐标系
4     save_dir: output
5 RECON_INIT:
6     camera_paths: [images/0000.png, images/0001.png] # 选择场景初始化使用的相机
7 RECON:
8     method: pnp # 方法: pnp or epipolar
9     visualize: True
10    merge_3d: False
11    normalize_epi: True # 是否对对极几何重建中的位移向量进行尺度对齐
12 ESTIMATOR:
13    alg: magsac # 仅用于计算基础矩阵, 以得到相机位姿, cv2.findFundamentalMat(p1, p2,alg)
14    ransac_params:
15        ransacReprojThreshold: 0.1
16        confidence: 0.99
17    extract:
18        contrast_thresh: 0.001 # 基于对比度的关键点选择阈值。较低的值会增加特征数量, 但会降低
19        稳定性。
20        edge_thresh: 10 # 消除关键点边缘响应的阈值, 值越低, 越容易忽略边缘附近的特征, 从而减少
21        边缘引起的不匹配
22        sigma: 1.6
23    match:
24        thres: 0.5
25    alg: None # ransac, magsac, None
26    ransac_params:
27        ransacReprojThreshold: 10
28        confidence: 0.99
29        tree: 7 # 配置索引, 密度树的数量为5
30        checks: 50 # 指定递归次数
31        flan_k: 2 # 最近邻的数量, =2, 表示寻找两个最近邻, 一般不改变该值
32 BA:
33     least_square_params:
34         method: trf # 优化算法: trf or lm
35         ftol: 1e-8 # 停止标准

```

简单起见，在之后的叙述中，如果没有特殊说明，我们都仅给出与上述设置相比有变更的参数设置，其余参数保持上述默认值。

2.2 SIFT: Features 提取

2.2.1 理论分析

SIFT (Scale Invariant Feature Transform, 尺度不变特征变换匹配算法) 是由 David G. Lowe [1] 在 1999 年提出的高效区域检测算法，在 2004 年 [2] 得以完善。SIFT 算法的主要流程为：

构建尺度空间 首先通过多尺度高斯滤波构建高斯金字塔：

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1)$$

其中，高斯核函数定义为：

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

高斯差分金字塔 通过计算相邻尺度的高斯图像差异构建高斯差分金字塔：

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (3)$$

关键点检测与定位 在高斯差分金字塔中搜索局部极值点，并通过计算主曲率去除边缘响应：

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (4)$$

关键点方向分配 计算关键点的梯度和方向，构建梯度方向直方图：

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (5)$$

$$\theta(x, y) = \arctan \left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \right) \quad (6)$$

描述子生成 最后，构建描述子，描述子通过梯度方向直方图来表征：

$$\text{描述子} = \{\text{HOG 特征}\} \quad (7)$$

总体而言，SIFT 算法通过在多个尺度上提取特征并生成描述子，有效地识别并描述了图像中的稳定局部特征。这为后续的特征匹配和三维重建任务打下基础。

2.2.2 实际实现

而在本次实验中，我们选用 OpenCV 库中的 `cv2.SIFT_create` 函数进行 SIFT 特征提取，其主要参数有：

- `contrastThreshold`: 基于对比度的关键点选择阈值。较低的值会增加特征数量，但会降低稳定性。

- `edgeThreshold`: 消除关键点边缘响应的阈值，值越低，越容易忽略边缘附近的特征，从而减少边缘引起的不匹配。
- `sigma`: 高斯滤波器的标准差 (standard deviation)，用于生成高斯金字塔的基础图像。默认值为 1.6。

在创建 SIFT 对象之后，我们通过 `sift.detectAndCompute` 函数得到关键点和描述子，并用于之后的特征匹配。代码如下：

```
1 def extract_features(image, contrast_thresh=0.04, edge_thresh=10, sigma=1.6,
2     image_name = "0000", save_dir = "output"):
3     # Convert the image to grayscale
4     gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
5
6     # Initialize the SIFT detector with specified thresholds
7     sift = cv2.SIFT_create(contrastThreshold=contrast_thresh, edgeThreshold=edge_thresh
8         , sigma = sigma)
9
10
11    # Detect keypoints and descriptors using SIFT
12    keypoints, descriptors = sift.detectAndCompute(gray_image, None)
13
14    # Draw keypoints on the image for visualization
15    image_with_keypoints = cv2.drawKeypoints(image, keypoints, None, flags=cv2.
16        DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
17
18    # Create a directory for saving output images if it doesn't exist
19    dir = os.path.join(save_dir, "sift_keypoints")
20    if not os.path.exists(dir):
21        os.makedirs(dir)
22    save_path = os.path.join(dir, f'{image_name}_sift_keypoints.png')
23    cv2.imwrite(save_path, image_with_keypoints)
24    return keypoints, descriptors, image_with_keypoints
```

2.3 图像特征匹配

图像特征匹配是计算机视觉中一项重要技术，用于在不同图像之间找到对应的特征点。本节将概述图像特征匹配的主要流程及其实现。

2.3.1 理论分析

1. 特征提取 首先，从每幅图像中提取特征点及其描述符。这些描述符能够唯一标识图像中的关键点，并在图像之间进行比较。这一步在上一章节中已经完成。

2. 特征匹配 使用近似最近邻搜索算法（例如 FLANN）来快速匹配不同图像的特征描述符。匹配过程中，通常采用 Lowe 的比率测试来筛选出好的匹配对：

$$\text{如果 } \frac{\text{distance}(m)}{\text{distance}(n)} < T, \text{ 则接受匹配} \quad (8)$$

其中， m 和 n 为最近邻和次近邻， T 为阈值。

3. 特征再筛选 利用 RANSAC (Random Sample Consensus) 或 MAGSAC (Marginalizing Sample Consensus) 算法进一步验证匹配的一致性，排除误匹配，从而提高匹配的鲁棒性。

2.3.2 实际实现

在实际实现中，我们使用 `cv2.FlannBasedMatcher` 进行 KNN 匹配。以下是特征匹配实现的示例代码摘要，展示了如何在实际应用中执行上述步骤：

```
1 def match_features(img1, img2, features1, features2, thres= 0.4, tree = 5, checks =
2     50, flan_k = 2):
3     # 特征匹配逻辑
4     # Create FLANN matcher object
5     FLANN_INDEX_KDTREE = 1 # 建立FLANN匹配器的参数
6     index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=tree)
7     search_params = dict(checks=checks) # or pass empty dictionary
8     flann = cv2.FlannBasedMatcher(index_params, search_params)
9     matches = flann.knnMatch(descriptors1, descriptors2, k=2)
10    # 应用Lowe比率测试
11    good_matches = [(m,n) for m, n in matches if m.distance < thres * n.distance]
12    img_matches = cv2.drawMatchesKnn(img1, keypoints1, img2, keypoints2, matches, None)
13    return img_matches
```

此处所用的几个参数为：

- `thres`: 用于判断匹配是否良好的 lowe 比率
- `tree`: 配置索引，密度树的数量。
- `checks`: 指定递归次数。
- `flan_k`: 最近邻的数量，=2 表示寻找两个最近邻，一般设置为 2。

注意，尽管在参数设置中我们提供了在匹配中启用再筛选算法的选项，但是在实际实现中，**我们并没有用 RANSAC 或 MAGSAC 等方法对特征进行再筛选**，因为在尝试后发现算法会剔除掉过多的匹配，导致重建效果中 3D 点数量过少，甚至出错等问题。因此，在默认参数设置中，我们令 `match.alg = None`。

2.4 对极几何场景初始化

2.4.1 理论分析

对极几何是用于理解两个摄像机视图之间几何关系的重要工具。通过对极几何，我们可以从两幅图像中估计出相机的运动和三维结构信息。以下步骤详细介绍了场景初始化的流程。

1. 计算基础矩阵和本质矩阵 首先，通过特征点匹配计算基础矩阵 F 和本质矩阵 E 。基础矩阵 F 是场景几何中的一个核心概念，它包含了两幅图像间的对极约束。本质矩阵 E 则是基础矩阵的一个特例，适用于已知内参的情况：

$$E = K^T F K$$

其中 K 是相机的内参矩阵。

2. 恢复相机姿态 从本质矩阵 E 中，我们可以恢复出相对旋转 R 和平移向量 t 。这可以通过对 E 进行奇异值分解 (SVD) 并使用以下关系得到：

$$E = U \Sigma V^T, \quad R = U W V^T \quad \text{或} \quad U W^T V^T, \quad t = U[:, 2]$$

其中 W 是一个特定的旋转矩阵。

3. 三角测量 利用恢复的相机姿态和内参矩阵，我们可以对匹配点进行三角测量，从而重建三维点坐标。三角测量的数学表达式为：

$$X = P_1^{-1} p_1 = P_2^{-1} p_2$$

其中 P_1 和 P_2 是从世界坐标到图像坐标的投影矩阵， p_1 和 p_2 是对应的图像点。

2.4.2 实际实现

在实现过程中，我们将 0000 号图像的相机设置为世界坐标系，并用 0000 和 0001 号图像进行场景初始化。在经过特征提取和特征匹配后，我们能根据匹配结果，得到两个图像下对应的 2D 点对。我们利用 `cv2.findFundamentalMat` 得到从 0000 到 0001 的基础矩阵 E 。在此基础上，继续使用 `cv2.recoverPose`，输入基础矩阵 E ，相机内参矩阵 K ，和对应的点对 `points1` (P_{0000})，`points2` (P_{0001})，我们即可得到 0001 的外参矩阵 $[R|t]$ ，其在点对之间起到了转换的作用：

$$P_{0001} = R P_{0000} + t \tag{9}$$

接着使用三角定位法 `cv2.triangulatePoints(P1, P2, points1, points2)` 得到初始的 3D 点。其中 $P1$, $P2$ 为两个相机的投影矩阵:

$$P1 = K_{3 \times 3} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{3 \times 4} \quad P2 = K_{3 \times 3} \begin{bmatrix} R & t \end{bmatrix}_{3 \times 4} \quad (10)$$

最后对得到的 4 维向量 `points_4d_hom` 去齐次化即可得到初始化的 3D 点:

```
points_3d = points_4d_hom[:3] / points_4d_hom[3]
```

2.5 使用 Perspective-n-Point 进行场景重建

2.5.1 理论分析

PnP (Perspective-n-Point) 是一种常用的场景重建方法，它利用已知的 3D 点和它们对应的 2D 图像点，估计相机的姿态（旋转和平移）。假设我们有一组 3D 点 $P_i = [X_i, Y_i, Z_i]^\top$ 和它们对应的 2D 图像点 $p_i = [u_i, v_i]^\top$ ，相机内参矩阵为 K 。PnP 问题可以表示为:

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad (11)$$

其中， s_i 是一个尺度因子， R 是 3×3 旋转矩阵， t 是 3×1 平移向量。PnP 的目标是估计 R 和 t 。

常见的 PnP 求解方法有:

1. P3P (Perspective-3-Point): 使用三个点求解，得到多个可能的解，然后利用第四个点进行消歧。
2. EPnP (Efficient PnP): 将 3D 点表示为四个虚拟控制点的加权和，将问题转化为估计这些控制点的坐标。
3. iterative 方法 (如 LM 优化): 将 PnP 问题表示为最小化重投影误差的非线性优化问题:

$$\min_{R,t} \sum_{i=1}^n \left\| p_i - \frac{1}{s_i} K[R|t] P_i \right\|^2 \quad (12)$$

然后使用 iterative 优化方法 (如 Levenberg-Marquardt 算法) 求解。

2.5.2 实际实现

PnP 的关键在于得到匹配的 2D-3D 点对。在使用 0000, 0001 号图像进行场景初始化后，我们可以得到两组 2D-3D 点对 pair_0 和 pair_1 。其中，两组点对的 3D 点是一样的。那么，在此基础上，我们可以对 0001 和 0002 进行特征提取和特征匹配，对于 0002 中和 0001 匹配的 2D 点，我们在 pair_1 找到对应的 3D 点，将 2D-3D 点对加入到 pair'_2 中。在得到 pair'_2 后，我们就可以使用 `cv2.solvePnP` 得到 0002 相对世界坐标系的外参矩阵，继而将 0001, 0002 相机的内、外参和对应的匹配点输入到三角定位函数中，得到重新定位的 3D 点。我们可以仅仅用三角定位得到的新的 3D 点求得 pair_2 ，也可以将其与 pair'_2 合并得到 pair_2 。对于 0003，我们重复上述的操作，这是一个迭代的过程，以下是伪代码：

Algorithm 1 基于 PnP 的场景重建

- 1: **初始化重建** (`init_recon`): 得到 img-0, img-1 的外参，得到 img1 的 2D-3D 点对
 - 2: **for** $i = 1$ **to** 10 **do**
 - 3: **图像特征提取**: 对 img- i , img- $(i + 1)$ 图像进行特征提取
 - 4: **特征匹配**: 对 img- i , img- $(i + 1)$ 进行特征匹配
 - 5: **建立 2D-3D 点对**: 在 pair_i 中找到与 img- $(i + 1)$ 匹配的 2D 点对应的 3D 点，得到 pair'_{i+1}
 - 6: **使用 PnP 求解外参**: 利用 `cv2.solvePnP` 得到 cam_{i+1} 的外参
 - 7: **三角定位**: 输入 cam_i , cam_{i+1} 内外参和匹配点，使用三角定位获取新的 3D 点和 2D-3D 点对
 - 8: **选择是否合并**: 可以选择只用三角定位得到的新的 3D 点计算得到 pair_{i+1} ，或者与 pair'_{i+1} 合并得到 pair_{i+1}
 - 9: **end for**
-

具体细节请参考实际代码 `src/pnp_recon.py`。

2.6 Bundle Adjustment 进行场景优化

2.6.1 理论分析

Bundle Adjustment (束调整) 是三维重建中的一个关键步骤，用于同时优化场景中三维点的坐标和摄像机的参数（如位置和方向），以最小化投影误差。以下是该过程的详细描述和必要的公式。

1. 投影模型 在 Bundle Adjustment 中，每个三维点通过摄像机的内参矩阵和外参矩阵投影到二维图像平面上。设摄像机的内参矩阵为 \mathbf{K} ，外参由旋转矩阵 \mathbf{R} 和平移向量 \mathbf{t} 组成，投影模型可以表示为：

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

其中, \mathbf{P} 是投影矩阵, \mathbf{K} 是内参矩阵, \mathbf{R} 是旋转矩阵, \mathbf{t} 是平移向量。

2. 投影过程 给定一个三维点 \mathbf{X} 和相应的摄像机参数, 其在图像平面上的投影 x 可以通过以下公式计算:

$$x = \mathbf{K}(\mathbf{R}\mathbf{X} + \mathbf{t})$$

将三维点和摄像机参数联合优化, 目标是最小化实际观测到的二维点和通过投影模型计算得到的二维点之间的误差。

3. 误差函数 误差函数是实际观测点和投影点之间差的欧氏距离, 给定 n 个摄像机和 m 个点, 误差函数可以表示为:

$$f(x) = \sum_{i=1}^n \sum_{j=1}^m \|x_{ij} - \mathbf{K}_i(\mathbf{R}_i \mathbf{X}_j + \mathbf{t}_i)\|^2$$

其中, x_{ij} 是第 i 个摄像机观测到的第 j 个点的图像坐标。

4. 最小化误差 使用非线性最小二乘方法来最小化误差函数。通常, 为了处理大规模问题, 会使用稀疏矩阵技术来提高计算效率。

2.6.2 实际实现

在实际实现中², 我们使用了 `scipy` 库的 `least_squares` 函数, 将 3D 点的坐标和 10 个相机 (去掉了作为世界坐标系的相机) 的外参矩阵作为优化参数, 将重投影误差写作目标点和重投影点之间的残差传入 `least_squares` 函数。

具体而言, 在计算重投影误差时, 我们需要使用各个相机的 2D-3D 点对 (来自 PnP); 此外, 我们需要确保优化的时候不会有同一个 3D 点在参数中出现多次的情况, 否则优化过后该点将“分裂”, 这是不合适的。所以我们需要对重复的 3D 点进行过滤, 并以维护下标的方式来处理不同相机中重复出现了 3D 点:

```
1 def build_dataset(data_path):
2     results_dict = np.load("...."), allow_pickle=True).item()
3     points_3d = [] # (n_points, 3)
4     pcolors = []
5     camera_indices = [] # (n_observations, )
6     point_indices = [] # (n_observations, )
7     points_2d = [] # (n_observations, 2)
8     camera_params = []
9     points_3d_set = set()
10    points_3d_index_map = {}
11    for img in results_dict.keys():
```

²参考: https://scipy-cookbook.readthedocs.io/items/bundle_adjustment.html

```

12     pair = results_dict[img]['pair_2D_3D']
13     pcolor_list = results_dict[img]['pcolor']
14     R = results_dict[img]['r']
15     t = results_dict[img]['t'].ravel()
16     rvec = cv2.Rodrigues(R)[0].ravel()
17     camera_params.append(np.hstack((rvec, t)))
18     # filter out the same 3D points
19     for i, p in tqdm(enumerate(pair), desc=f"Processing {img}"):
20         if tuple(p[1]) not in points_3d_set:
21             points_3d_set.add(tuple(p[1]))
22             points_3d.append(p[1])
23             pccolors.append(pcolor_list[i])
24             index = len(points_3d) - 1
25             points_3d_index_map[tuple(p[1])] = index
26             points_2d.append(p[0]) # we dont care the repeat of 2d points, since it's
27             # not included in params
28             camera_indices.append(int(img) - 1)
29             point_indices.append(index)
30     else:
31         points_2d.append(p[0])
32         camera_indices.append(int(img) - 1)
33         point_indices.append(points_3d_index_map[tuple(p[1])])
33 ..... 省略更多

```

此外，我们还使用了 `least_square` 支持的稀疏矩阵，通过指定雅可比矩阵中非零元素的位置来加速了计算。具体细节请参照 `src/bundle_adjustment.py`

3 结果

3.1 特征提取展示

图 2 展示了默认参数设置下，使用 SIFT 算法提取的特征点。特征点较小，请放大查看。

3.2 特征匹配展示

图 3 展示了默认参数设置下，使用 FlanMatcher+MAGSAC 得到的匹配特征

3.3 场景初始化

默认参数设置下前两张图间的相机 pose (外参矩阵)。最终结果请以提交的.txt 文件为准。



图 2: 默认参数设置下, 使用 SIFT 算法提取的特征点

$$[\mathbf{R} \mid \mathbf{t}] = \left[\begin{array}{ccc|c} 0.988247 & 0.025328 & 0.150750 & 0.997435 \\ -0.022479 & 0.999536 & -0.020573 & 0.020882 \\ -0.151201 & 0.016942 & 0.988358 & -0.068462 \end{array} \right]$$

3.4 PnP 结果展示

图 4 展示了默认参数设置下, PnP 的 3D 重建结果展示。其中图右侧的部分离群点对应图像中的右侧墙面及建筑。

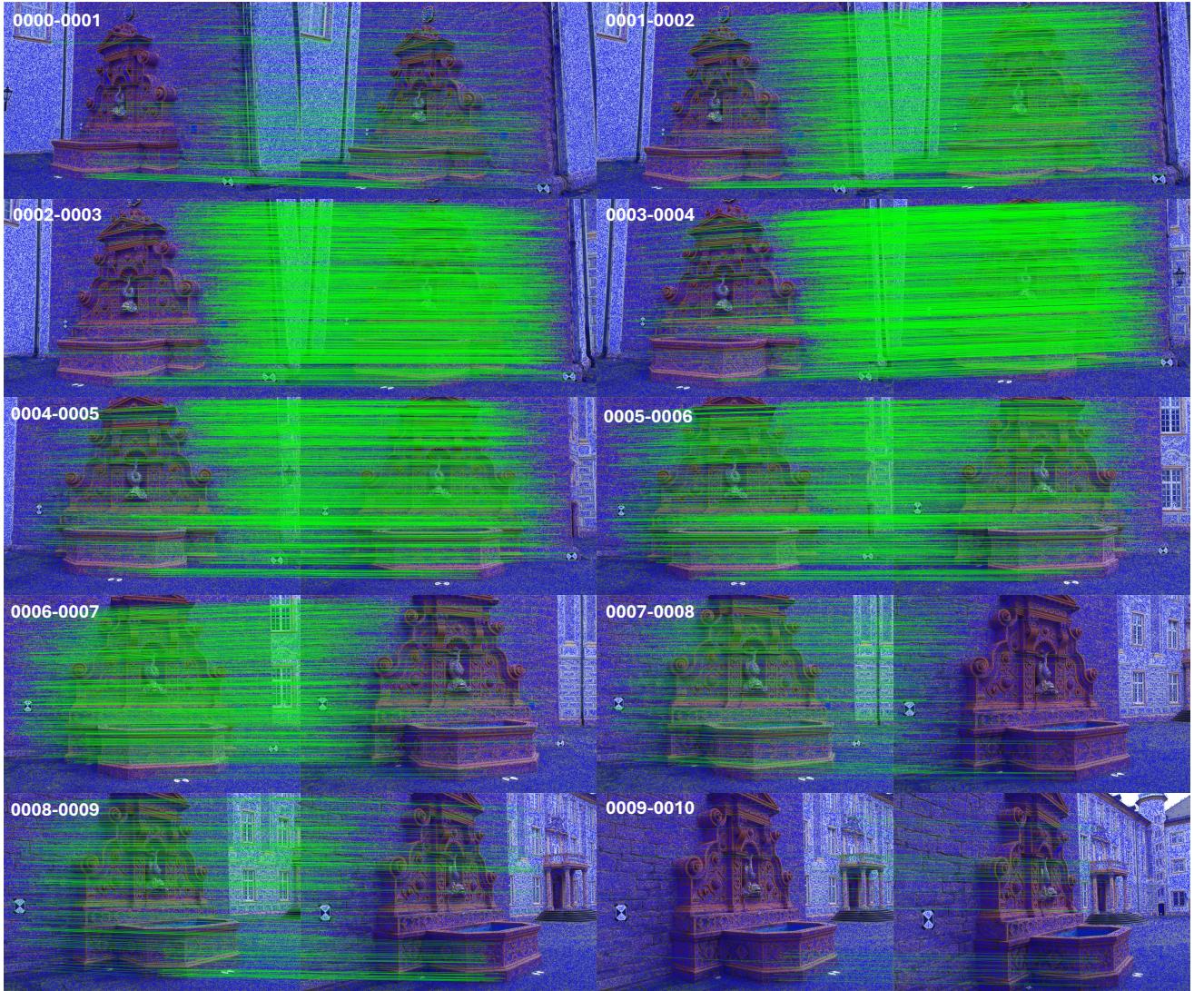


图 3: 默认参数设置下, 使用 FlanMatcher+MAGSAC 得到的匹配特征

3.5 Bundle Adjustment 结果展示

图 5 展示了在默认参数设置下, 对 PnP 方法返回的结果进行 Bundle Adjustment 优化后最终的 3D 重建结果。就图上比较而言, 肉眼较难看出优化前后的区别。

4 讨论

4.1 不使用 Bundle Adjustment

图 6 是 Bundle Adjustment 前后的重投影误差对比。尽管从图 4, 5 中难以看出两者的显著差别, 但是从重投影误差来看, BA 优化前后其数量级有显著的改变。BA 算法并不能增加点云中的点云数量, 但是就减少重投影误差而言, 其显然能够提升点云、相机位姿的精度。在实验中, 无论哪种参数设置, BA 只要能成功优化, 其优化后的重投影误差基本都约等于 0, 这提示我们应该去调整参数以收集尽可能多的点, 直到 BA 算

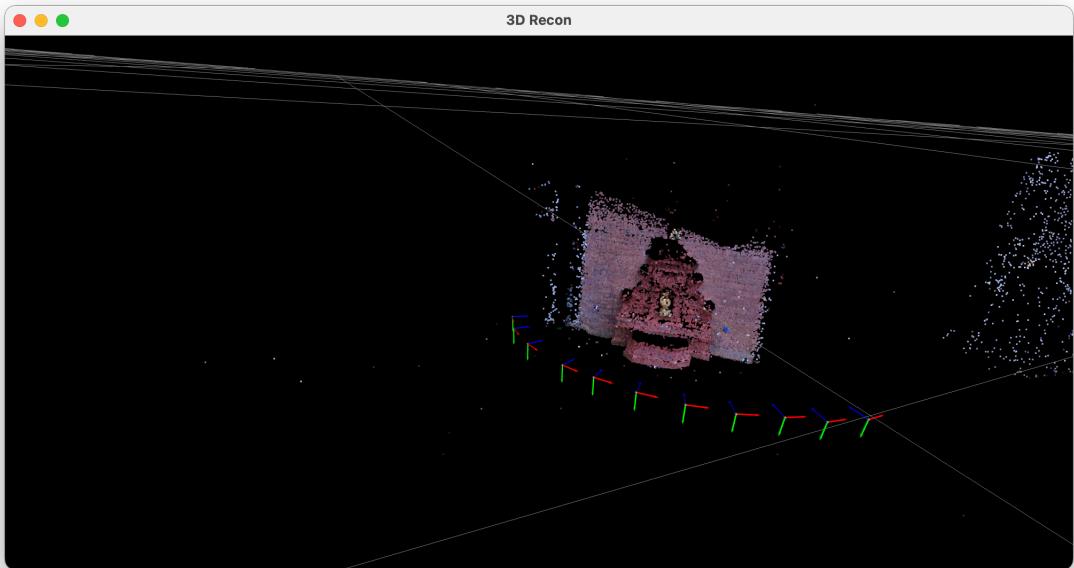


图 4: 默认参数设置下, PnP 的 3D 重建结果展示

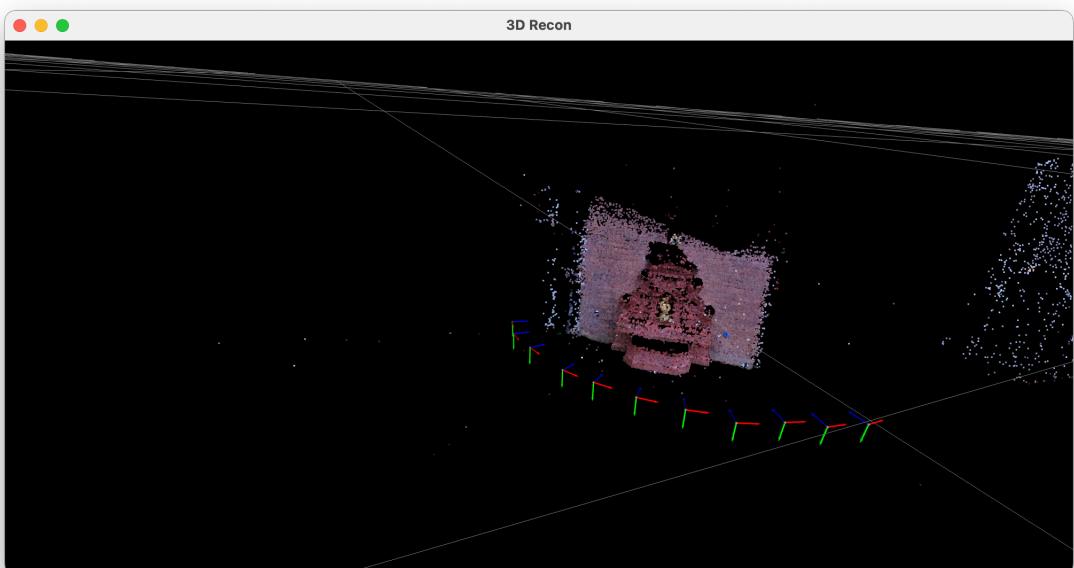


图 5: 默认参数设置下, 经过 Bundle Adjustment 优化后的 3D 重建结果展示

法无法优化（误差过大、反常点过多等）。

4.2 对特征点提取中阈值参数选取的分析

在这一章节中, 我们使用控制变量的方法来探究改变参数中的 `contrast_thresh` 和 `edge_thresh` 对使用 PnP 方法重建得到的结果的影响。在实验中, 我们保持其他参

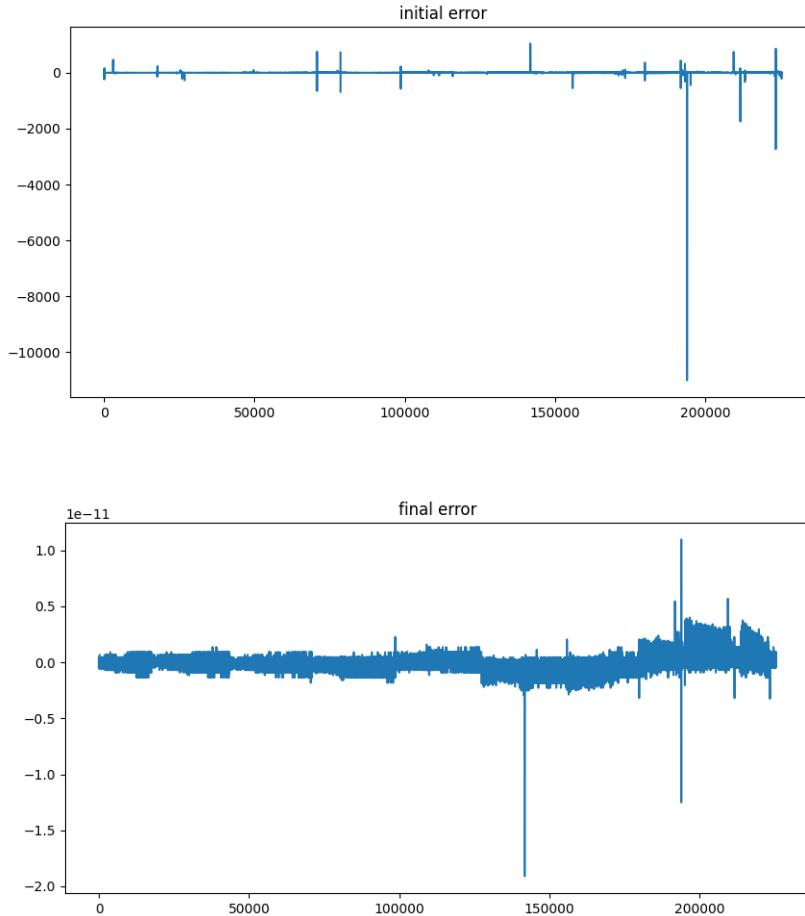


图 6: Bundle Adjustment 优化前后的重投影误差。图中的 y 轴的值是残差, 来自 `(points_proj - points_2d).ravel()`。横轴则是各个 2D 点的下标, 由于使用了 `ravel`, 所以 x 和 y 被展开了, 实际点数为 `indexNumber / 2`

数设置为默认值。

contrast_thresh 该参数为对比度阈值³, 用于过滤低对比度区域中的特征点。阈值越大, 检测器产生的特征越少。从表格中可以看出, 随着 `contrast_thresh` 的增加, 初始重投影误差的大小呈下降趋势, 而得到的 3D 点数目整体呈下降趋势, 但我们发现 $1e-4$ 的点数要小于 $1e-3$ 处。经过比较两种设置下的程序日志, 我们发现每张图片的关键点数目为 $1e-4$ 略大于 $1e-3$, 但图片对之间“优良匹配”的数目则并没有这种单调的规律。考虑到我们使用了 FlanKNNMatch, 当 `contrast_thresh` 足够小导致每张图片关键点数目近乎相同时, $1e-4$ 比 $1e-3$ 多引入的关键点可以被视作一种噪声, 从而在实际匹配过程中未能提供有价值的信息, 甚至可能导致匹配质量的降低。这解释了为何在 $1e-4$ 设置下虽然关键点数量稍多, 但优良匹配的数目并未明显增加。因此, 虽然降低 `contrast_thresh` 可以在理论上增加检测到的关键点数, 但并非所有这些新增点都是有用的, 有可能导致

³<https://www.cnblogs.com/silence-cho/p/15170216.html>

表 1: 对特征点提取中 `contrast_thresh` 参数取值的分析

<code>contrast_thresh</code>	Points Number	Reprojection Error	BA Optimized	Success
1e-4	105904	4.9285e+08	4.0379e-20	✓
1e-3	106064	7.0393e+07	3.8484e-20	✓
1e-2	96651	5.4572e+06	3.3056e-20	✓
1e-1	399	4.0768e+05	2.0222e-22	✓

引入更多误匹配使匹配质量下降，经过过滤后得到的“优良匹配”数目反而降低，进而影响最终的重建质量。

edge_thresh 该参数用于过滤掉类似图片边界处特征的阈值（边缘效应产生的特征），其值越大，检测器产生的特征越多（过滤掉的特征越少）。

表 2: 对特征点提取中 `edge_thresh` 参数取值的分析

<code>edge_thresh</code>	Points Number	Reprojection Error	BA Optimized	Success
9	102912	2.1002e+07	4.5632e-15	✓
10	106064	7.0393e+07	3.8484e-20	✓
11	108253	2.9798e+09	2.2285e-19	✓
12	110032	5.9653e+11	2.5990e-08	✓

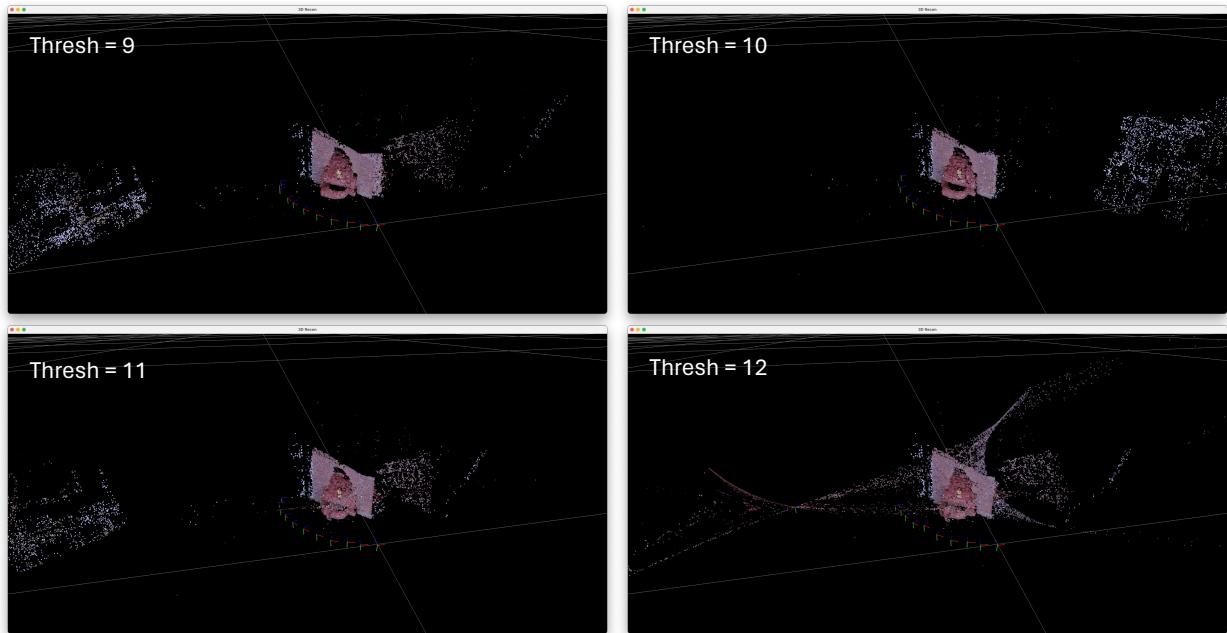


图 7: 使用不同 EdgeThresh 得到的重建结果 (Bundle Adjustment 之后)

4.3 对特征点匹配中阈值参数选取的分析

在这一章节中，我们探究改变参数中 `match.thres`, 也即公式 8 中的 T 参数对使用 PnP 方法重建得到的结果的影响。在实验中，我们保持其他参数设置为默认值。

表 3: 对特征点匹配中 `match.thres` 参数取值的分析

Match Thres	Points Number	Reprojection Error	BA Optimized	Success
0.1	-	-	-	✗
0.3	21342	1.4975e+03	5.0288e-21	✓
0.4	60737	1.5147e+06	2.6226e-20	✓
0.5	106064	7.0393e+07	3.8484e-20	✓
0.7	181591	1.0252e+16	-	✗

从表 3 中结果可以看出，当 `thres` 取值过小的时候 (0.1)，由于匹配数量过小 (甚至为 0)，导致程序报错。而当 `thres` 过大时 (0.7)，由于引入过多异常匹配，重投影误差无法通过 BA 优化。

4.4 对 `cv2.findFundamentalMat(p1, p2, alg)` 中算法选择的探究

表 4: 基础矩阵计算中的算法选择分析

Algorithm	Points Number	Reprojection Error	BA Optimized	Reconstruction Success
RANSAC	106064	8.1600e+07	3.8158e-20	✓
MAGSAC	106064	7.0393e+07	3.8484e-20	✓

从表 4 的结果中可以看出，在其余参数保持默认的情况下，RANSAC 算法和 MAGSAC 算法对重建结果的影响并不是很大。在提交的结果中，我们选取了 BA 优化后最终重投影误差更小的、来自 RANSAC 的结果。

4.5 仅使用对极几何进行三维重建

在初始化场景中，我们使用了对极几何得到了 `cam0001` 相对于 `cam0000`(世界坐标系) 的 pose；同样的，我们可以继续使用对极几何，来得到 `cam0002` 相对于 `cam0001` 的 pose，如此反复，我们可以得到所有相机的 pose。而通过三角定位法，我们自然可以在每一次相对 pose 的计算中得到一组 3D 点，将这组 3D 点进行聚合，就可以重建 3D 场景。

通过这种方法，我们得到了图中的结果。可以看到，3D 场景的重建并不成功，与 PnP + BA 相比，其出现了明显的“错位”。分析来看，有以下两点可能的原因。

尺度因子没有对齐 仅仅用对极几何重建缺少对齐的**尺度因子**。以下是理论推导：

给定两幅图像中的匹配点 (x_1, y_1) 和 (x_2, y_2) ，它们对应于归一化的相机坐标系中的点 \mathbf{x}_1 和 \mathbf{x}_2 。两个相机的相对运动可以用旋转矩阵 R 和平移向量 t 来描述。如果我们将 t 转换为一个斜对称矩阵 t_\times ，则本质矩阵 E 可以表示为：

$$E = Rt_\times$$

这里， t_\times 是 t 的斜对称矩阵形式，用于表示叉积，使得 $t_\times \mathbf{v} = \mathbf{t} \times \mathbf{v}$ 对于任意向量 \mathbf{v} 。具体而言，任意向量 $\mathbf{v} = (v_1, v_2, v_3)$ 的反对称矩阵 \mathbf{v}_\times 定义为：

$$\mathbf{v}_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

这个矩阵有一个关键性质：对于任意向量 \mathbf{u} ，矩阵乘法 $\mathbf{v}_\times \mathbf{u}$ 等价于向量 \mathbf{v} 与向量 \mathbf{u} 的叉乘：

$$\mathbf{v}_\times \mathbf{u} = \mathbf{v} \times \mathbf{u}$$

我们回到验证过程，对极几何的极线约束表示为：

$$\mathbf{x}_2^T E \mathbf{x}_1 = 0$$

这表示，第二幅图像中的点 \mathbf{x}_2 和第一幅图像中的点 \mathbf{x}_1 必须满足上述方程。本质矩阵 E 定义了从第一相机到第二相机的相对旋转和平移。因为 E 是由 R 和 t_\times 构成，我们可以看到 E 与平移向量 t 的长度无关。具体来说，如果我们将 t 缩放一个非零常数 λ ，即 $t' = \lambda t$ ，则 t'_\times 会变为 λt_\times 。新的本质矩阵 E' 会是：

$$E' = R(\lambda t_\times) = \lambda(Rt_\times) = \lambda E$$

极线约束依然成立，因为：

$$\mathbf{x}_2^T (\lambda E) \mathbf{x}_1 = \lambda (\mathbf{x}_2^T E \mathbf{x}_1) = \lambda \cdot 0 = 0$$

因此，无论 λ 取何值（只要非零），极线约束都满足，说明从图像数据只能确定 t 的方向，而不能确定其大小。这表明在单目视觉里，除非有外部信息或其他约束（如尺度信息），否则无法从两幅图像中恢复出绝对的平移尺度。

但经过检查，我发现 opencv 库中使用 recoverPose 函数得到的 t 的 norm 值基本都为 1，如果我们取 cam_1 相对世界坐标系下的平移向量作为尺度，由于其值为 1，后续得到的相机的位移向量并不需要进行放缩。所以，我们认为**尺度因子对齐并不是对极几何重建的误差的主要来源**。

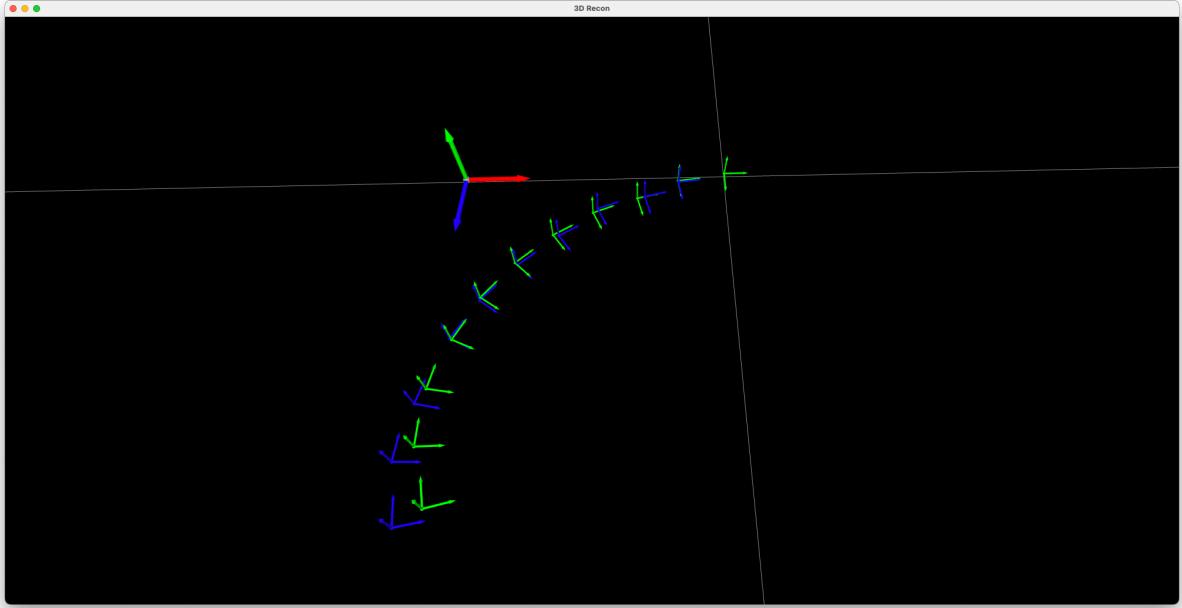


图 8: 比较 PnP + BA(blue) 与 Epipolar(green) 重建的相机位姿结果

误差累积 由于我们在计算 cam_{i+1} 的外参矩阵是基于 cam_i 的，那么 cam_i 的误差自然会传递到 cam_{i+1} 的计算中，且会随着迭代的进行误差不断变大。图 8 展示了这一现象：此外，我们给出逐个相机比较的旋转矩阵精度和平移相机精度：

$$\text{旋转矩阵精度: } \frac{\|\mathbf{R}_{\text{pred}}^\top \mathbf{R}_{\text{GT}} - \mathbf{I}\|_2}{\|\mathbf{I}\|_2}, \quad (13)$$

$$\text{角度差异: } \text{acos} \left(\frac{\text{trace}(\mathbf{R}_1 \mathbf{R}_2^\top) - 1}{2} \right), \quad (14)$$

$$\text{平移向量精度: } \frac{\|\mathbf{t}_{\text{GT}} - \mathbf{t}_{\text{pred}}\|_1}{\|\mathbf{t}_{\text{pred}}\|_1} \quad (15)$$

图 9 展示了在默认参数设置下，比较以 PnP + BA 方法作为基准，使用对极几何重建得到的相机位姿的差异。我们发现，其误差基本从 cam0 到 cam10 逐渐增加，且比较符合图 8 的趋势，这进一步验证了对极几何重建效果差是由于误差累积的假设。

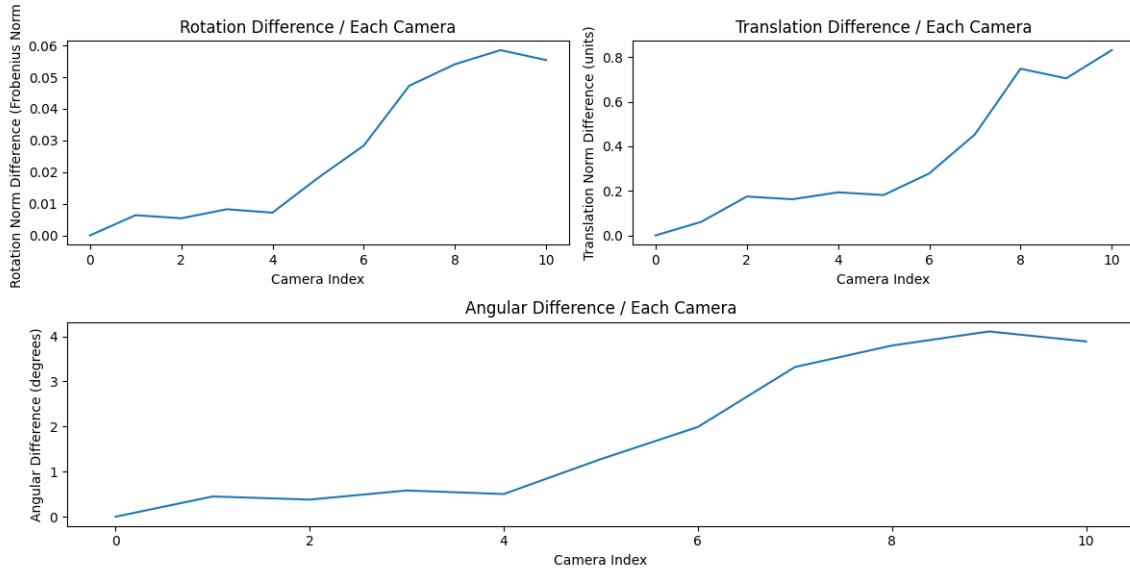


图 9: 在默认参数设置下, 比较以 PnP + BA 方法作为基准, 使用对极几何重建得到的相机位姿的差异

5 总结

在此作业中, 我们成功实现了一个基于传统计算机视觉技术的三维重建流程。通过综合运用尺度不变特征变换 (SIFT) 算法进行特征提取、FLANN 匹配器结合 Lowe 比率测试进行特征匹配、对极几何方法和 PnP 算法进行场景初始化和重建, 以及光束平差法 (Bundle Adjustment) 进行场景优化, 我们显著提高了三维重建的质量和精度。

实验结果表明, SIFT 算法在特征提取阶段对于图像中关键特征的识别具有重要作用, 而特征匹配的准确性直接影响了后续的三维重建效果。通过对极几何方法和 PnP 算法的应用, 我们能够较为准确地恢复相机姿态并重建出初始的三维点坐标。此外, Bundle Adjustment 作为后处理步骤, 通过最小化重投影误差, 进一步优化了三维点坐标和相机参数, 使得重建结果更为精细。

在参数选择方面, 我们发现 SIFT 算法中的对比度阈值 (`contrast_thresh`) 和边缘阈值 (`edge_thresh`) 的选取对特征点的数量和质量有显著影响。实验表明, 适当的阈值选择可以有效提升特征匹配的稳定性和准确性。同时, 我们也探究了特征匹配中的匹配阈值 (`match.thres`) 对重建结果的影响, 发现适当的匹配阈值能够平衡匹配数量和质量, 避免引入过多异常匹配。

此外, 我们还对不同的算法选择 (如 RANSAC 与 MAGSAC) 在计算基础矩阵时的影响进行了分析, 结果表明这些算法在本研究的上下文中对最终的重建结果影响不大。

然而, 当去除 Bundle Adjustment 模块, 仅使用对极几何方法进行三维重建时, 我们观察到了误差累积和尺度因子对齐问题, 这限制了该方法的应用。

总体而言, 本次作业加深了我们对三维重建技术的理解, 培养了我对计算机视觉领域的探索兴趣, 我收获颇丰。

最后，万分感谢助教学长在我完成本次作业的过程中给予我的莫大帮助！

参考文献

- [1] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
- [2] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.