作业 运动恢复结构与光束法平差

姓名：陈福浩

学号：2112504215

1. 简介

在本作业中，我实现了一个增量式运动恢复结构 (SfM) 管道，旨在从一组 2D 图像重建 3D 场景。该管道包括特征检测、匹配、使用 RANSAC 进行几何验证、场景图构建以及增量重建。

2. 实现细节

2.1 预处理 (`preprocess.py`)

关键点检测 (`detect_keypoints`)

我使用了 SIFT (尺度不变特征变换) 算法来检测图像中的关键点并计算描述符。

- **实现：** 我调用了 `cv2.SIFT_create()` (或为了兼容性使用 `cv2.xfeatures2d.SIFT_create()`) 来初始化 SIFT 检测器，并使用 `detectAndCompute` 获取关键点和描述符 。

特征匹配 (`create_feature_matches`)

为了寻找图像对之间的对应关系，我使用了基于 k-最近邻 (KNN) 的暴力匹配 (Brute-Force Matching)。

- **实现：** 使用 `cv2.BFMatcher` 为每个描述符找到前两个最佳匹配。

  过滤： 应用了 Lowe's ratio test (比例为 0.75) 来过滤模糊匹配。只有当最近邻距离显著小于次近邻距离时，才保留该匹配 。

几何验证 (`create_ransac_matches`)

为了从特征匹配中剔除外点 (outliers)，我在 RANSAC (随机抽样一致性) 循环中使用了 5 点算法来估计本质矩阵 (Essential Matrix) 。

- **实现：** 使用 `cv2.findEssentialMat` 和 `cv2.RANSAC` 方法。该函数计算本质矩阵并返回一个掩码，指示哪些点满足对极几何约束。
- **修改说明：** 这里的关键是使用了 `cv2.findEssentialMat` 而不是 `cv2.findEssentialMatrix`，以匹配当前 OpenCV 版本的正确函数签名。

场景图构建 (`create_scene_graph`)

构建了一个场景图，其中节点代表图像，边代表经过几何验证的匹配。

实现： 遍历所有图像对，如果某一对图像经过 RANSAC 步骤后拥有足够的内点 (由 `min_num_inliers` 决定)，则在图中添加一条边 。

2.2 运动恢复结构 (`sfm.py`)

初始化 (`get_init_image_ids` & `get_init_extrinsics`)

- **图像对选择 (`get_init_image_ids`):** 初始图像对的选择对重建质量至关重要。最初的朴素策略是选择内点数最多的一对，但这在密集数据集 (如 Temple) 上会导致选择相邻帧 (如帧 1 和帧 2)，它们的基线 (baseline) 太小，导致三角化深度不准确，重建结果扁平化。

  改进策略: 我修改了选择逻辑，优先选择 **间隔适中 (3-15帧)** 且 **内点数最多** 的图片对。这确保了初始对既有足够的重叠进行匹配，又有足够的视差 (baseline) 进行准确的三角化 。

  位姿估计 (`get_init_extrinsics`): 使用 `cv2.recoverPose` 从预处理步骤计算出的本质矩阵中恢复旋转 (R) 和平移 (t)。这确定了第二张图像相对于第一张图像 (设为单位阵) 的位姿 。

增量重建 (`incremental_sfm`)

下一张图像选择 (`get_next_pair`): 算法迭代地选择下一张最佳图像进行注册。选择标准是：该图像必须尚未注册，且与已注册图像集拥有最多的内点匹配。

- **PnP 求解 (`solve_pnp`):** 为了注册新图像，需要求解 Perspective-n-Point (PnP) 问题。

  改进策略: 我使用了 `cv2.solvePnP` 并配合 `cv2.SOLVEPNP_EPNP` 标志。相比于默认的 `ITERATIVE` 方法，EPNP (Efficient PnP) 不需要初始猜测，且在点数较少或初始状态不佳时更稳健，不易陷入局部最优 (导致点云“压扁”或成线状) 。

  鲁棒性增强: 我添加了一个检查：如果有效的 2D-3D 对应点少于 6 个，则优雅地停止重建，防止程序崩溃并保存已有的结果。

  重投影误差 (`get_reprojection_residuals`): 计算观测到的 2D 点与投影回图像平面的 3D 点之间的欧氏距离，用于评估位姿估计的质量 。

  三角化 (`add_points3d`): 利用新注册的图像和已有的注册图像，通过 `cv2.triangulatePoints` 三角化出新的 3D 场景点 。

2.3 光束法平差 (bundle_adjustment.py)

残差计算 (compute_ba_residuals): 为了优化相机参数和 3D 点，我们需要最小化重投影误差。

向量化实现: 根据作业要求，我实现了**无循环 (No loops)** 的残差计算。利用 Numpy 的矩阵操作 (`np.matmul`, `np.concatenate`)，一次性将所有观测点对应的 3D 坐标投影回相机平面并计算欧氏距离。这极大地提高了优化速度 。

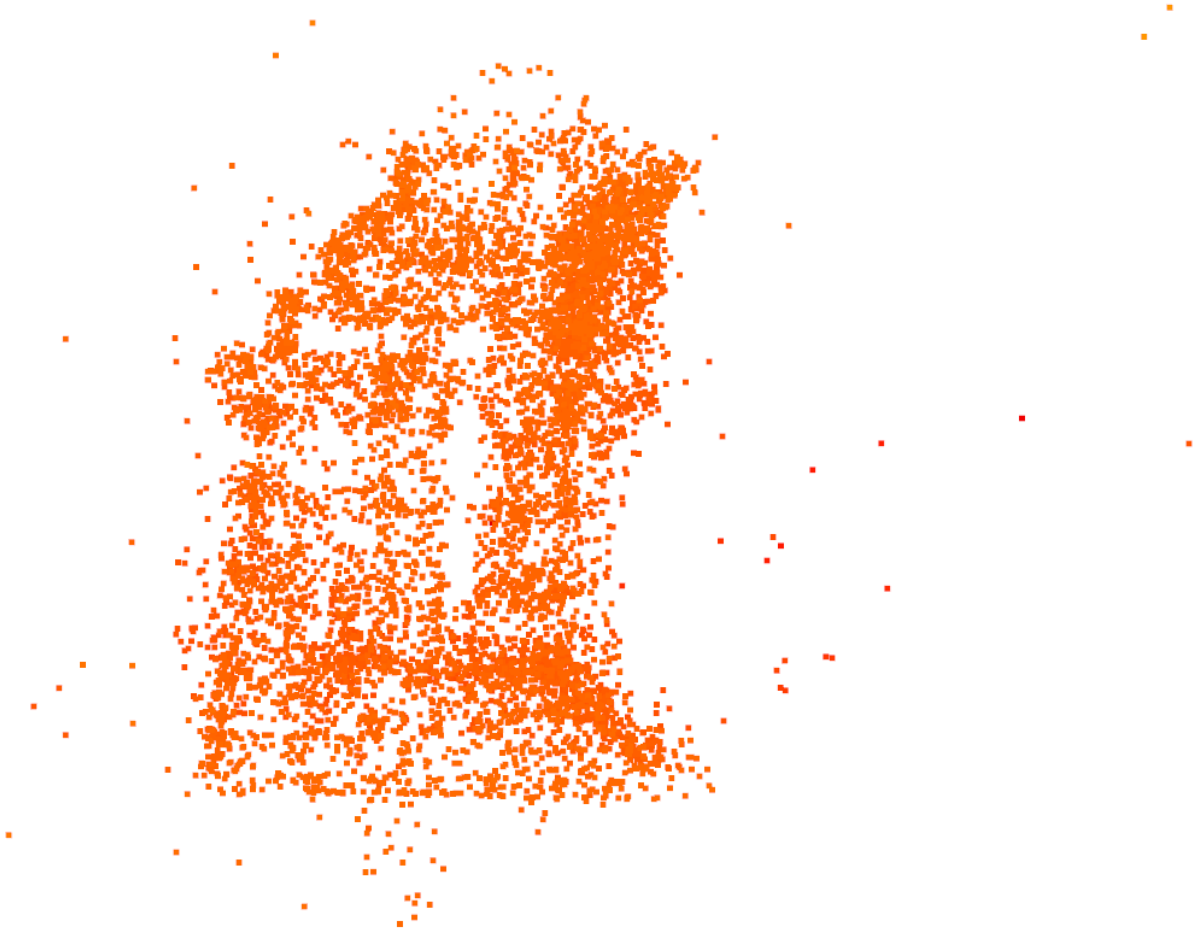
3. 实验结果

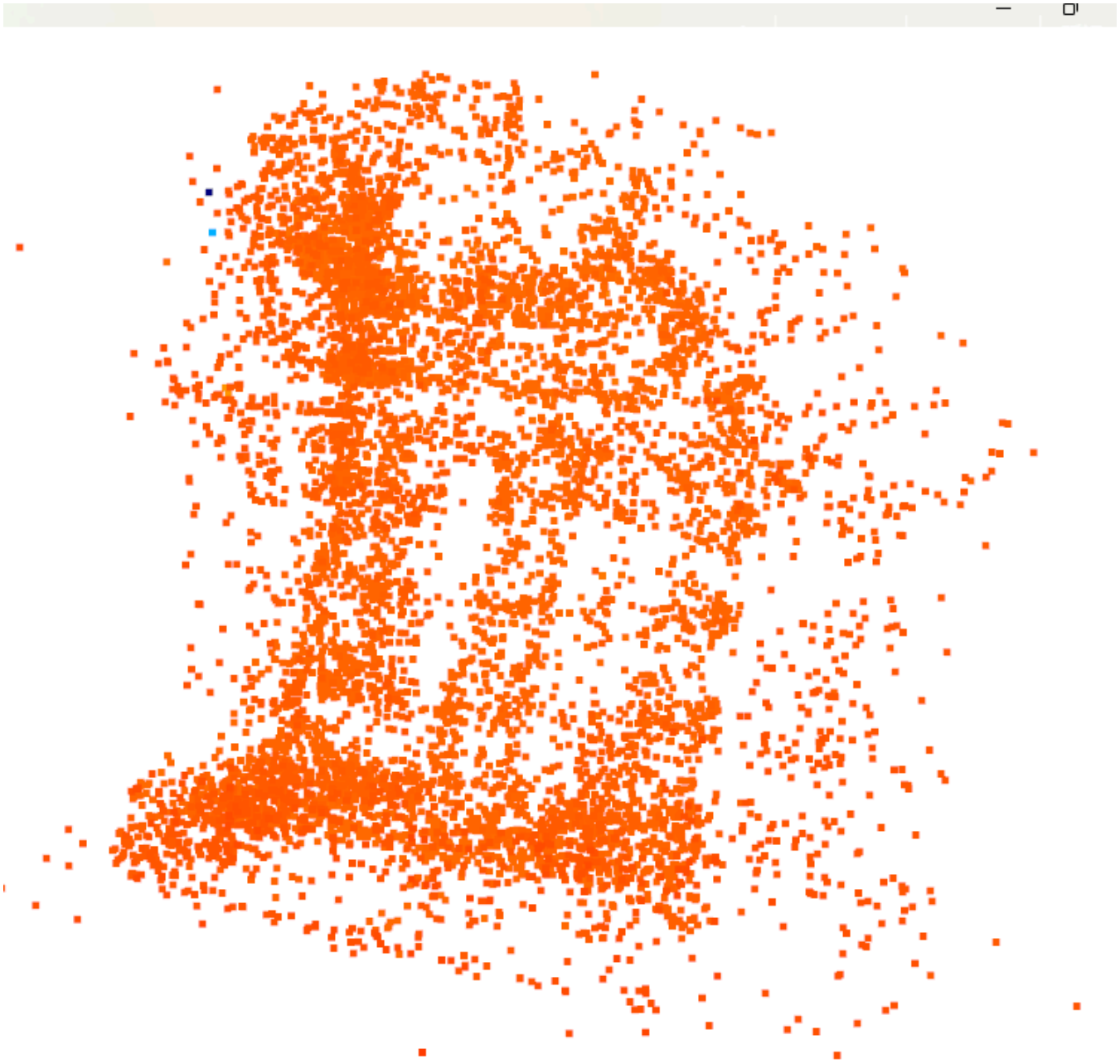
3.1 定性结果 (可视化)

以下是 `temple` 数据集的 3D 点云重建可视化结果。

Temple 的最终 3D 重建模型：

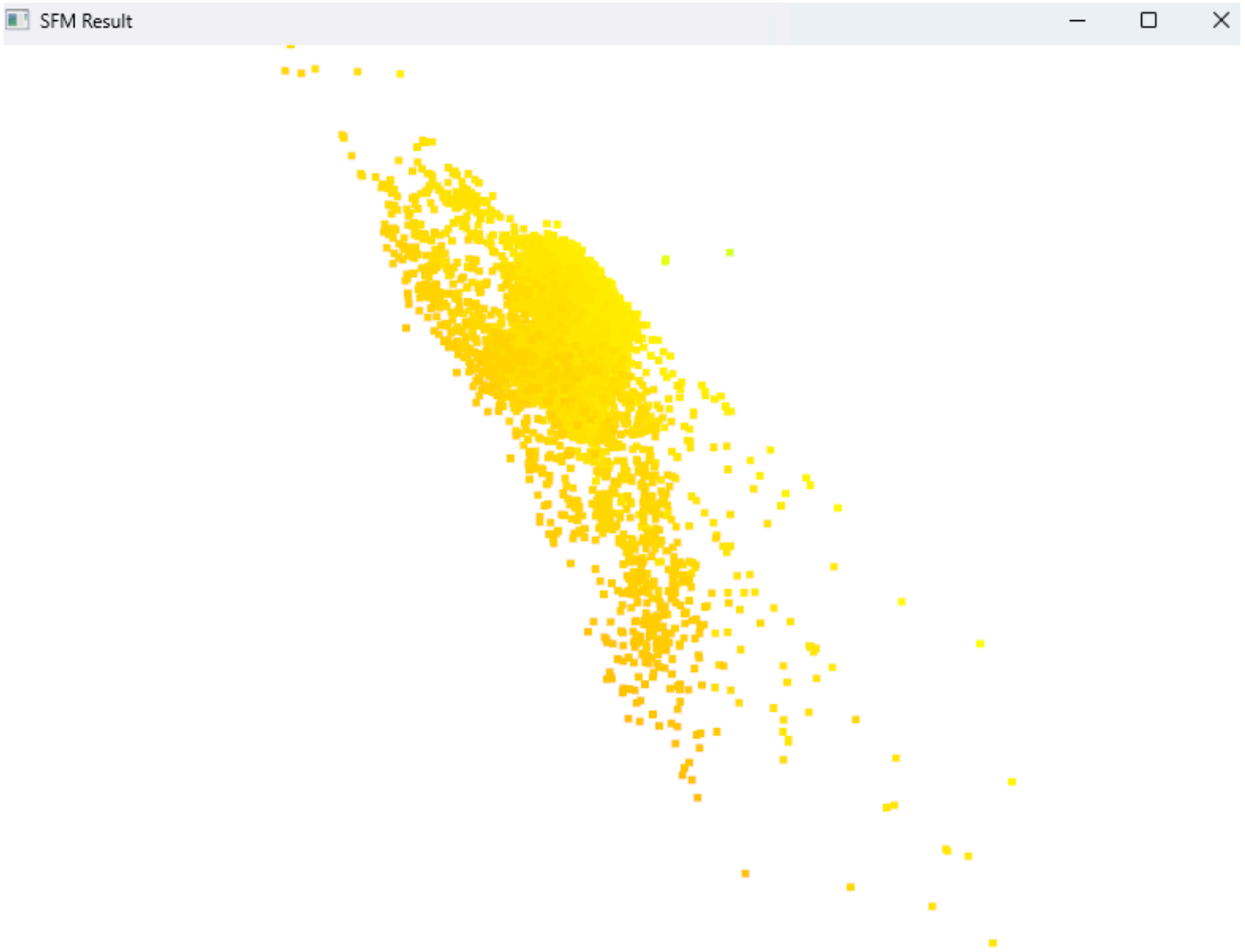
可以看到清晰的柱子和顶部的横梁结构，与作业描述中的预期输出相符





失败案例分析 (基线退化):

以下是失败的结果，图像被压缩成一个椭圆形没有任何轮廓，因为初始化基线过短导致的失败重建。由于选择了相邻帧作为初始对，三角化误差过大，导致结构塌缩成扁平状。


SFM Result

Mini-Temple 数据集优化：

3.2 定量结果

- **注册图像总数：** 45/47
- **总 3D 点数：** 8050
- 重建成功恢复了神庙的几何结构，柱子和横梁的点云簇清晰可辨。

4. 总结

在实验过程中，遇到的主要挑战是 **“小基线退化” (small baseline degeneracy)** 问题。最初，算法选择相邻帧 (如帧 1 和 2) 作为种子对，因为它们匹配点最多。然而，过小的基线导致三角化误差极大，重建结果塌缩成一条线。通过在初始化阶段强制施加最小帧间隔约束，并改用更稳健的 `EPnP` 求解器，系统最终成功恢复了正确的 3D 结构。此外，为了解决 Open3D 在包含中文路径的系统上崩溃的问题，还通过代码调整兼容了旧版渲染引擎。