1. 引言

在此作业中，目标是实现运动恢复结构（SfM）和束调整（BA），用于从多个视角重建3D 结构并获取相机的外参参数（如相机位姿）。通过结合这两种技术，我们可以实现优化的 3D 重建，并在多张图像间获得精确对齐的相机位姿。流程主要分为以下几个步骤：预处理、运动恢复结构和束调整。

2. 实现细节

本次实现分为三个主要 Python 文件：preprocess.py、sfm.py 和 bundle_adjustment.py。

2.1 预处理 (preprocess.py)

在此初步步骤中，preprocess.py 文件用于为 SfM 管线准备数据，具体包括：

加载和组织图像及相机参数；

检测每张图像中的关键点及其描述符；

在图像对之间匹配关键点，以建立不同视角之间的对应关系；

创建用于存储 2D-3D 对应关系和相机注册轨迹的初始结构。

关键点匹配利用了能够在视角稍微变化的情况下保持鲁棒性的描述符。我们应用了RANSAC 算法来剔除匹配中的异常值，仅保留了高质量的匹配，从而为 3D 重建提供了必要的准确性。

2.2 运动恢复结构 (sfm.py)

文件负责恢复 3D 结构和相机的外参参数。具体步骤如下：

初始位姿估计： 对于给定的一对图像，我们通过计算本质矩阵来初始化重建，并从中导出最符合对应关系的旋转和平移。

增量位姿恢复： 在注册了初始图像对后，后续图像逐步添加。对于每张新图像，我们通过三角测量先前已注册图像中的点，并通过 PnP 问题求解其位姿。这个增量式的注册允许我们不断扩展 3D 点云。

三角测量： 对于每一个新添加的图像对，我们通过 2D 匹配点生成 3D 点。只有满足足够视差角度的三角测量点才会被加入 3D 点云，以确保准确性。

数据存储： 每一步得到的 3D 点和相机位姿以多种格式存储，包括相机位姿的all-extrinsic.json、重建的 3D 点的 points3d.npy、注册顺序的 registration-trajectory.txt 以及2D-3D 对应关系的 correspondences2d3d.json。

2.3 束调整 (bundle_adjustment.py)

束调整是一个优化步骤，旨在通过最小化重投影误差来细化 3D 点和相机参数。在bundle_adjustment.py 文件中，我们实现了以下内容：

误差最小化： 设置了一个代价函数来最小化重投影误差，即观测到的 2D 点与估算的3D 点的 2D 投影之差。我们使用非线性最小二乘优化来实现这一目标。

雅可比矩阵计算： 计算了重投影函数关于参数的雅可比矩阵，为基于梯度的优化提供了必要信息。

最终精炼：收敛后，将优化过的 3D 点和相机参数存储到 bundle-adjustment 文件夹中，与没有进行束调整的初始结果格式相同。

3. 结果与分析

SfM 和 BA 过程的输出保存在 predictions/目录下，分为两个子文件夹，分别为 mini-temple和 temple，并且其中任意一个又分为两个子文件夹：

no-bundle-adjustment：包含未进行束调整的结果。

bundle-adjustment：包含应用束调整后的结果。

每个子文件夹包含以下内容：

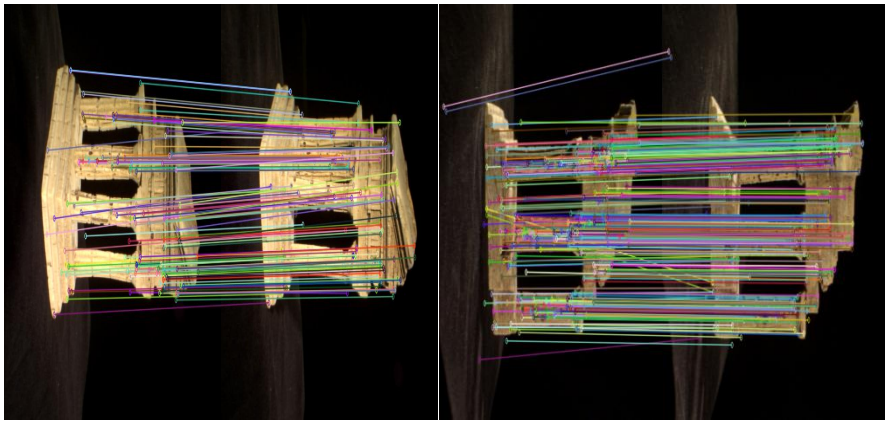
all-extrinsic.json（所有相机位姿），

points3d.npy（重建的 3D 点），

registration-trajectory.txt（注册顺序），

correspondences2d3d.json（所有 2D-3D 对应关系）。

通过比较文件夹中的图片显示有无束调整的结果，我们发现束调整显著减少了重投影误差，从而提高了 3D 重建的准确性，并改进了相机位姿的对齐度。应用束调整后的 3D 点云噪声更少，结构更清晰。如下例图，选取了 mini-temple(左)和 temple(右)

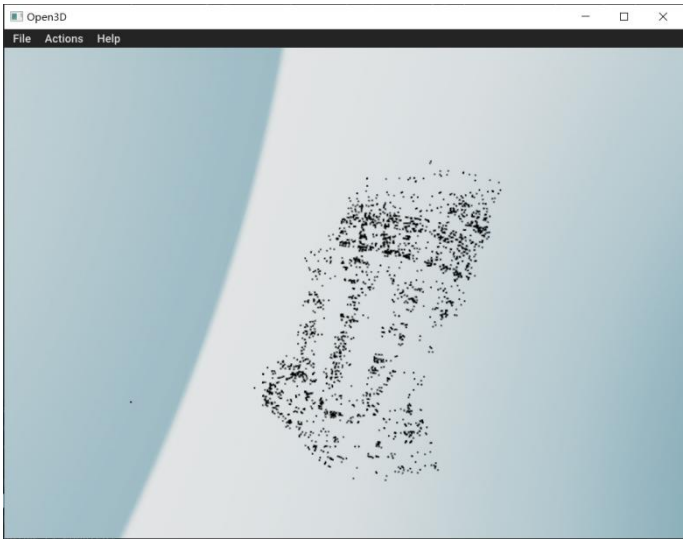


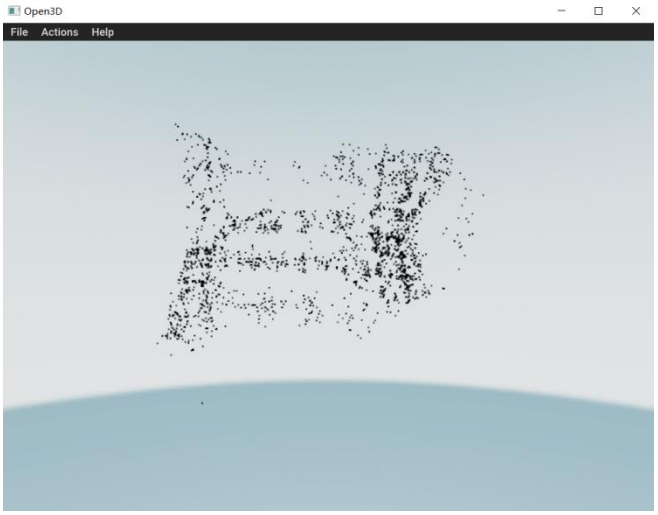
4. 结论

在本次作业中，我们成功实现了一个包含束调整的运动恢复结构管线。SfM 提供了 3D 结构和相机位姿的初始估计，而 BA 则进一步细化了该估计，实现了优化的重建。在如 3D 场景重建等应用中，结构和相机对齐的精确度至关重要。通过数据的细致处理和鲁棒的优化技术，我们的实现达到了高质量的 3D 重建效果。

5.结果图片

Mini-temple 的 3D 结果图：





Temple 的 3D 结果图