

Report

1、预处理 preprocess.py

主要用于执行运动结构 (SfM) 算法的预处理步骤。这个阶段的目标是从图像中检测关键点，匹配特征点，并为后续的三维重建和相机参数估计做好准备。

- 关键点检测

脚本使用了 SIFT (Scale-Invariant Feature Transform) 算法来检测图像中的关键点。SIFT 特征具有尺度 and 旋转不变性，非常适合 SfM 任务中处理不同视角和光照变化的情况。具体实现中，首先通过 `cv2.SIFT_create()` 初始化 SIFT 检测器，然后使用 `detectAndCompute()` 来提取图像中的关键点和描述符。

- 特征匹配

在特征匹配部分，使用了 OpenCV 提供的暴力匹配器 (BFMatcher) 进行特征匹配。为了提高匹配的准确性，采用了 Lowe 比例测试来过滤错误匹配，确保保留下来的特征点配对具有较高的可靠性。匹配后的结果保存在一个 `.npy` 文件中，用于后续的几何验证。

- 几何验证

为了确保特征匹配的几何一致性，使用了 RANSAC 算法来计算两个图像之间的本质矩阵。通过 RANSAC 进行鲁棒估计，可以去除错误匹配的离群点，确保最终的特征匹配对能够用于精确的相机姿态估计。

- 输出

在完成上述步骤后，`preprocess.py` 会将关键点及其描述符存储到 `.pkl` 文件中，同时保存特征匹配结果和经过 RANSAC 验证后的匹配对。这些输出将为后续的三维点重建和相机参数的优化提供基础。

```
(leaf) PS C:\Users\yuh\Desktop\homework-04-Lab03-sfm-ba-viper333-master> python preprocess.py --dataset temple
INFO: detecting image keypoints...
INFO: creating pairwise matches between images...
INFO: creating ransac matches...
INFO: creating scene graph...

(leaf) PS C:\Users\yuh\Desktop\homework-04-Lab03-sfm-ba-viper333-master> python preprocess.py --dataset mini-temple
INFO: detecting image keypoints...
INFO: creating pairwise matches between images...
INFO: creating ransac matches...
INFO: creating scene graph...
```

2、sfm.py 代码

- 初始化

初始化阶段通过选择图像对来计算相对姿态。选择的图像对是基于最大数量的RANSAC内点来确定的。选定的图像对用于计算其外参矩阵，包括旋转矩阵和位移向量。此外，通过三角测量 (Triangulation) 获取初始的三维点集。

- 三维点重建

通过图像对中的匹配点，计算出三维点。这些匹配点通过图像的相机内参和外参矩阵进行投影，得到三维空间中的点云。

- 增量更新

增量更新阶段通过不断注册新的图像，将新图像与已注册图像进行匹配，并通过PnP (Perspective-n-Point) 方法估计新图像的相对位姿。每当一个新图像被成功注册时，新的三维点会被重建，并与现有的三维点进行更新。此过程通过逐步连接图像对，确保场景重建的连贯性。

- **光束调整**

在整个增量式SFM过程中, 为了优化相机的外参和三维点, 采用了光束调整方法。光束调整通过最小化重投影误差, 优化相机位姿和三维点的估计结果, 进一步提高了重建的准确性。

get_init_image_ids

此函数通过遍历场景图，选择具有最多匹配内点的图像对作为初始化图像对。

get_init_extrinsics

假设初始化图像1的外参为 $[I | 0]$ ，并通过计算图像对的本征矩阵，得到图像2的外参 $[R | t]$ 。

- **triangulate****

通过三角测量方法，计算出给定图像对和匹配点的三维坐标。

- **solve_pnp**

使用RANSAC算法解算PnP问题，估计出新的图像的旋转矩阵和位移向量。

- **bundle_adjustment**

该函数通过最小化光束调整残差，优化相机的内外参以及三维点的坐标，确保所有图像和三维点的相对位置更加精确。

```
(sfn) PS C:\Users\yuh\Desktop\homework-04-lab03-sfn-ba-viper3333-master> python sfn.py --dataset mini-temple
100% | 45/45 [00:04:00.80, 9.591t/s]

(sfn) PS C:\Users\yuh\Desktop\homework-04-lab03-sfn-ba-viper3333-master> python sfn.py --dataset mini-temple --ba
Iteration   Total_nflow   Cost   Cost reduction   Step norm   Optimality
1          12      5.3862e+02      6.76e-01      5.44e-04      2.31e+04
2          14      5.3350e+02      9.94e+00      2.72e-04      1.97e+04
3          15      5.3346e+02      1.20e-01      2.72e-04      4.66e+04
4          16      5.3633e+02      3.13e+00      6.80e-05      1.45e+04
5          17      5.2984e+02      1.29e+00      1.36e-04      1.57e+04
6          18      5.2854e+02      4.80e-01      1.35e-04      2.25e+04
7          19      5.2745e+02      9.87e-01      3.40e-05      1.08e+04
8          20      5.2694e+02      7.14e-01      6.80e-05      7.09e+03
9          21      5.2645e+02      4.88e-01      6.80e-05      1.73e+04
10         22      5.2684e+02      4.88e-01      6.80e-05      2.92e+04
11         23      5.2654e+02      4.70e-01      6.80e-05      2.76e+04
12         24      5.2690e+02      5.26e-01      6.80e-05      2.95e+04
13         25      5.2454e+02      4.82e-01      6.80e-05      3.02e+04
14         26      5.2403e+02      5.26e-01      6.80e-05      2.99e+04
15         27      5.2355e+02      4.76e-01      6.80e-05      3.06e+04
16         28      5.2355e+02      4.76e-01      6.80e-05      3.06e+04
17         28      5.2394e+02      5.10e-01      6.80e-05      3.00e+04
18         29      5.2257e+02      4.48e-01      6.80e-05      3.05e+04
19         30      5.2284e+02      5.10e-01      6.80e-05      3.01e+04
Function evaluations 30, initial cost 6.0598e+02, final cost 5.2204e+02, first-order optimality 3.01e+04.

(sfn) PS C:\Users\yuh\Desktop\homework-04-lab03-sfn-ba-viper3333-master> python sfn.py --dataset mini-temple
100% | 9/9 [00:01:00.80, 8.391t/s]
```

