Final project: *Visual SLAM*

Group member:

Yao Zhou;
Shuchong Wang;
Zhiyu Zhu;
Tianrun Yan;
Qiming Huang

Date: Dec. 10th, 2022

Abstract

ORB-SLAM is a feature-based monocular simultaneous localization and mapping (SLAM) system that operates in small and large indoor and outdoor environments. The system is robust to severe motion clutter, allows wide baseline loop closing and relocalization, and includes full automatic initialization. Building on excellent algorithms of recent years, we designed from scratch a novel system that uses the same features for all SLAM tasks: tracking, mapping, relocalization, and loop closing. A survival of the fittest strategy used to select points and keyframes of the reconstruction leads to excellent robustness. This strategy generates a compact and trackable map that only grows if the scene content changes, allowing lifelong operation. ORB-SLAM achieves unprecedented performance with respect to other state-of-the-art monocular SLAM approaches. We make the source code public for the benefit of the community.

Keywords：ORB-SLAM, mapping, keyframe, tracking

Catalog

Introduction

ORB-SLAM is a versatile and accurate SLAM solution for monocular, stereo and RGB-D cameras. It is able to compute in real-time the camera trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments, ranging from small hand-held sequences of a desk to a car driven around several city blocks. It is able to close large loops and perform global relocalization in real time and from wide baselines. Initialization from planar and non-planar scenes is automatic and robust. Below are videos, code, and related publications.

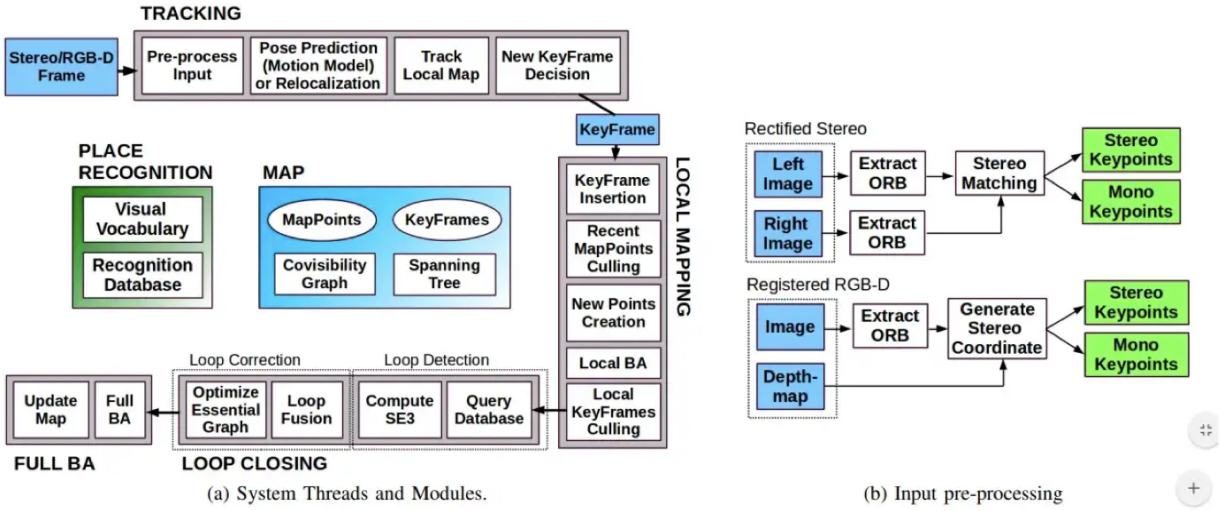

Figure 1: ORB-SLAM process map

Principle

Map Auto-initilization:

I. Searching points pair

The process would pick the eigenvalue of ORB from current keyframe. If it could not find this pair then it would do it again.

$$keyframe\ F_r\ ,\ x_c = x_r$$

II. Calculating parallel models（Hcr & Fcr）

$$x_c = H_{cr} x_r \qquad x_c^T F_{cr} x_r = 0$$

III. Calculating orientation

$$M_{00} = \sum_{X=-R}^{R} \cdot \sum_{Y=-R}^{R} I(x, y)$$

$$M_{10} = \sum_{X=-R}^{R} \cdot \sum_{Y=-R}^{R} x \cdot I(x, y) \qquad M_{01} = \sum_{X=-R}^{R} \cdot \sum_{Y=-R}^{R} y \cdot I(x, y)$$

$$Q_X = \frac{M_{10}}{M_{00}},\ Q_Y = \frac{M_{01}}{M_{00}}$$

$$\mathrm{C} = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) \quad \theta = \mathrm{atan2}(M_{01}, M_{10})$$

Tracking

we describe the steps of the tracking thread that are performed with every frame from the camera. The camera pose optimizations, mentioned in several steps, consist in motion-only BA, which is described in the Appendix.

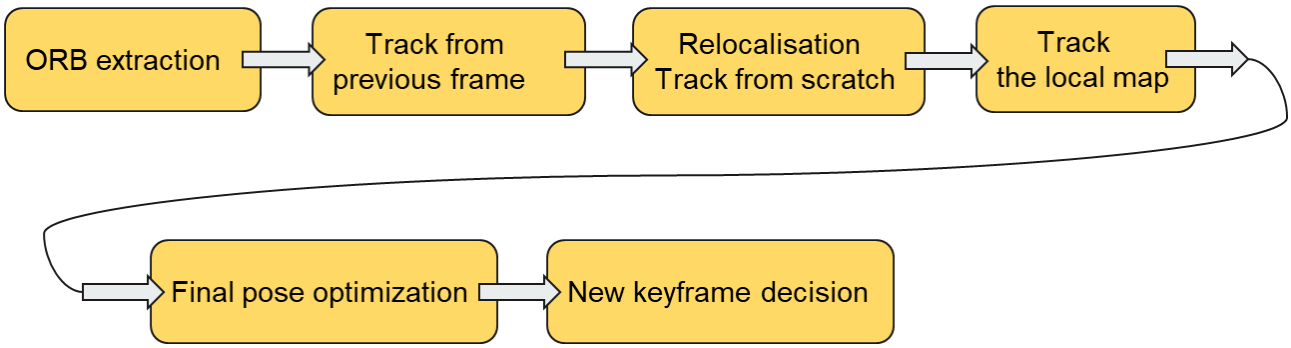

Figure 2: Tracking process

Local Mapping

For the local mapping part, the system mainly includes keyframe insertion, recent mappoints and local keyframes culling, new points creation as well as local bundle adjustment.

After the agent starts moving along a path while creating a new map or using a pre-saved one, part of the SLAM technique is localization. The ORB-SLAM3 library constantly checks similarities between extracted features from previous and current frames and executes localization in the map. If you start the system by creating a new map, the agent can just start moving down the path that later should be saved. On theother hand, if you choose to load an old atlas file to the system and it is done successfully, the agent needs to wait for relocalization before moving forwards.

Thereafter, a new map is created on top of an old one.If the relocalization is successful, the agent is placed on an old map - a newly created map still exists,

however, an active map changes and now an old map becomes the active one. The agent can now move and be localized in the old map.

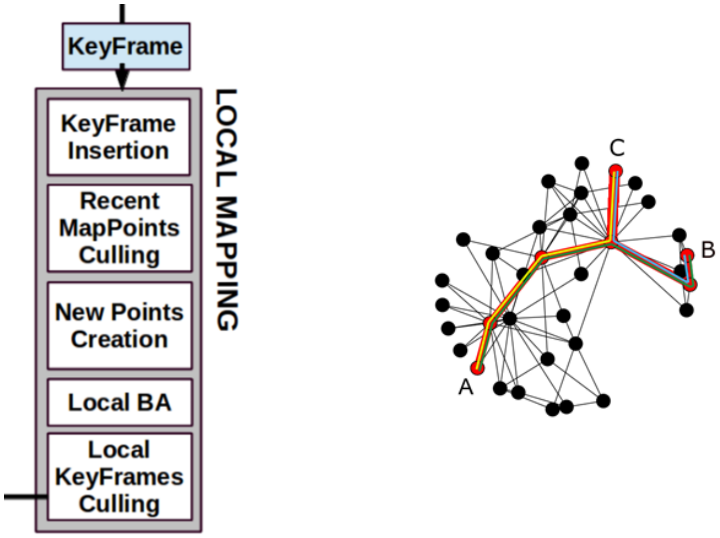

Figure 3: Local mapping

In addition to localization, another important feature is merging of maps. If you want to go from point A to point B and record it to a map, then you want to go from point A to point C and also save it to a map. One solution is to move in a way that ORB-SLAM3 library is able to merge paths. Consequently, if you wanted to record a map that would hold paths from A to B and from A to C, all you would need to do is move from point A to either B or C, slowly turn around and move towards the other one, and then back to point A. What should be saved, if the maps are correctly merged, is a red path, just like the right image shows.

As we can see, besides the map creations, once ORB-SLAM3 recognised the latest image as the one that was already seen, the possibility to merge maps was detected and it was performed successfully. When maps are merged, the recent map or

maps are deleted, the only remaining map is the one to which the recent map or maps are merged to.

Loop Closing

Every new keyframe loop closing is checked with the existing maps in Atlas. If found, pose-graph optimization is performed by using the Levenberg-Marquardt algorithm of an Essential Graph over similarity constraints (Sim3) , containing all the keyframes as nodes, but less edges than a normal covisibility graph.

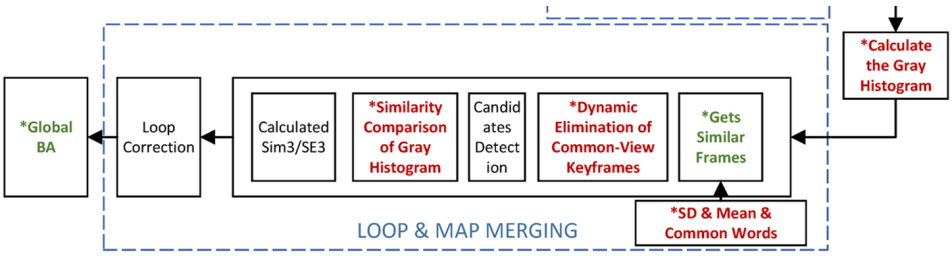

Figure 4: Loop closing

This graph is built from a spanning tree maintained by the system, and the loop closure procedure links strong edges from the covisibility graph. This graph variation has limited links, and keeps only the ones based on features visible from multiple keyframes.

Examined Dataset

The dataset examined for this project are EuRoC dataset and TUM dataset. The EuRoC dataset is a visual-inertial dataset that is collected on-board a Micro Aerial Vehicle (MAV). This dataset contains stereo images, synchronized IMU measurements, accurate motion and structure ground-truth. The EuRoC dataset is recorded with two pinhole cameras and an inertial sensor.

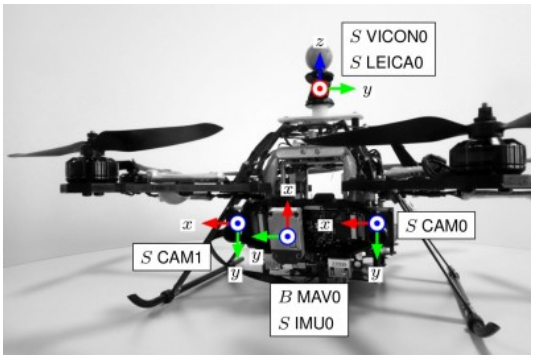

Figure 5: Kits

For the dataset collected in this project. The dataset was recorded by mobile phone, the dataset contains video.

Result:

In the left window, the green line is the real time trajectory of the camera lens. All the green dots in the right window are interest points. The result observed works well in the algorithm.

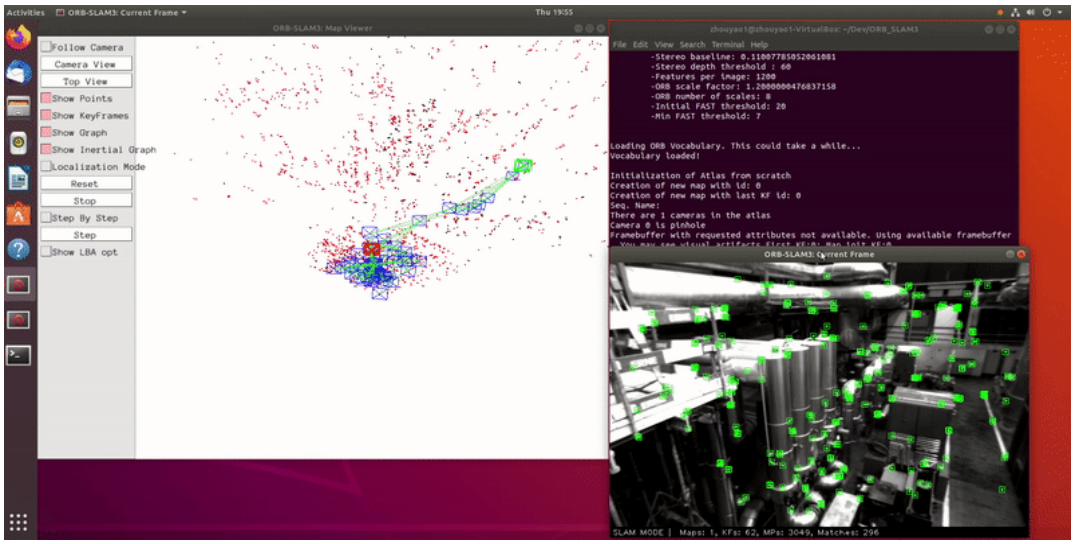

Figure 6: Applying the dataset into ORB-SLAM3 algorithm

The second dataset is the TUM dataset. This dataset is an RGB-D dataset. In addition, this dataset was recorded with two fisheye cameras and an inertial sensor. The dataset contains color and depth images of a Microsoft Kinect sensor along the ground truth trajectory of the sensor. After applying the dataset into ORB-SLAM3 algorithm. The result is shown below.

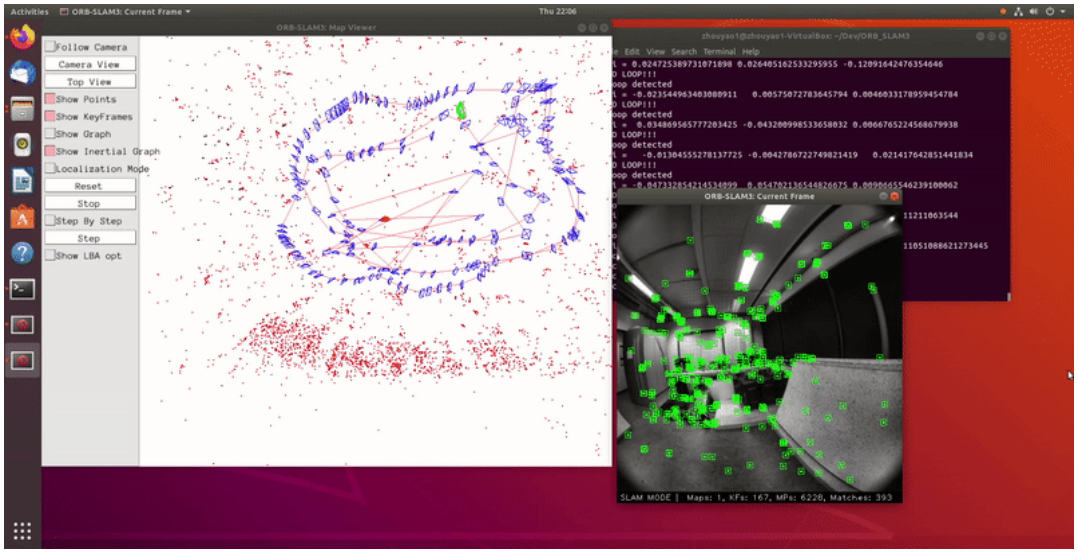

Figure 7: Applying the dataset into ORB-SLAM3 algorithm

Kitti:

The third dataset is the Kitti dataset. This dataset is recorded on rural highways. This dataset uses grayscale images. It has a much larger range of data. ORB_SLAM3 has a good processing effect on this data set。It can be seen from the trajectory diagram that the result is in line with the driving path of the vehicle. The disadvantage is that ORB_SLAM requires a large amount of memory for it, and track tracking fails several times due to insufficient memory.

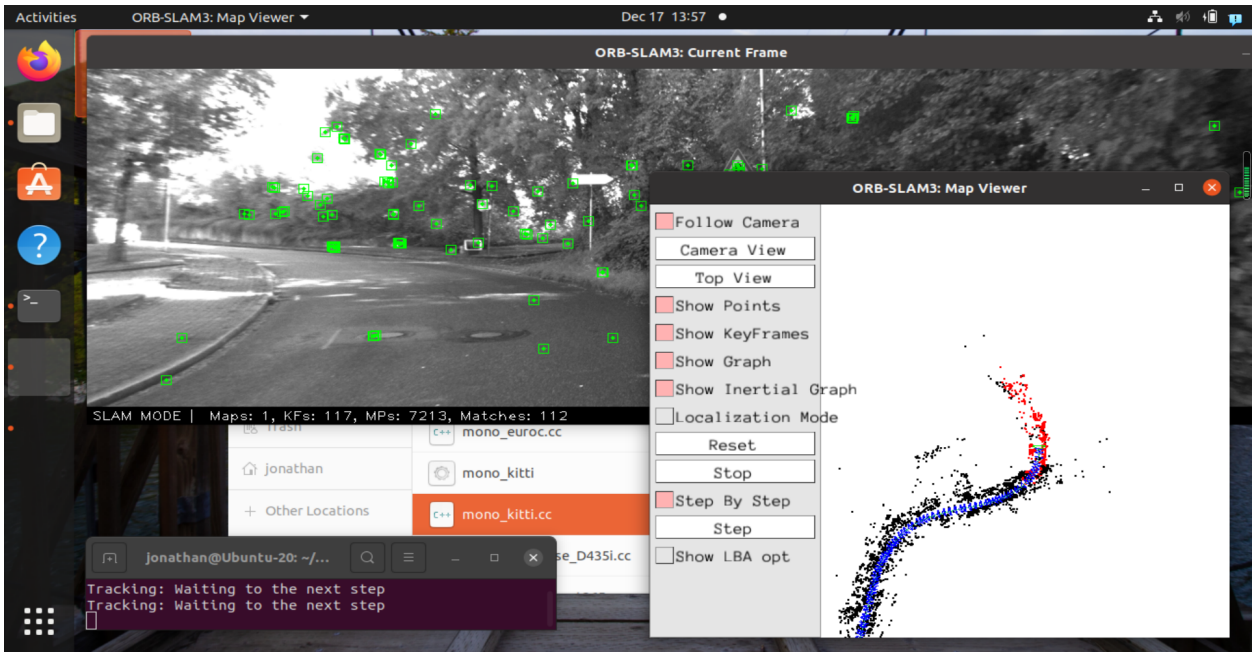

Figure 8: Applying Kitti into ORB-SLAM3 algorithm

Reference

[1] Rau´l Mur-Artal, J. M. M. Montiel, and Juan D. Tardo´s, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. IEEE TRANSACTIONS ON ROBOTICS, VOL. 31, NO. 5, OCTOBER 2015 1147.