8、ORB_SLAM2 basics

Official website： http://webdiis.unizar.es/~raulmur/orbslam/

ASL Dataset： https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets

mono Dataset： https://vision.in.tum.de/data/datasets/rgbd-dataset/download

stereo Dataset： http://robotics.ethz.ch/~asl-datasets/ijrr_euroc_mav_dataset/machine_hall/MH_01_easy/

orb_slam2_ros： http://wiki.ros.org/orb_slam2_ros

ORB-SLAM： https://github.com/raulmur/ORB_SLAM

ORB-SLAM2： https://github.com/raulmur/ORB_SLAM2

ORB-SLAM3： https://github.com/UZ-SLAMLab/ORB_SLAM3

8.1. Introduction

ORB-SLAM is mainly used for monocular SLAM;

ORB-SLAM2 version supports monocular, binocular and RGBD interfaces;

ORB-SLAM3 version adds IMU coupling and supports fisheye cameras.

All steps of ORB-SLAM use the ORB features of the image uniformly. The ORB feature is a very fast feature extraction method that is rotation invariant and can use pyramids to build scale invariance. Using unified ORB features helps the SLAM algorithm to be consistent in steps such as feature extraction and tracking, key frame selection, three-dimensional reconstruction, and closed-loop detection. The system is also robust to severe motion and supports wide-baseline closed-loop detection and relocalization, including fully automatic initialization. Since the ORB-SLAM system is a SLAM system based on feature points, it can calculate the camera's trajectory in real time and generate sparse three-dimensional reconstruction results of the scene.

Based on ORB-SLAM, ORB-SLAM2's contribution points are:

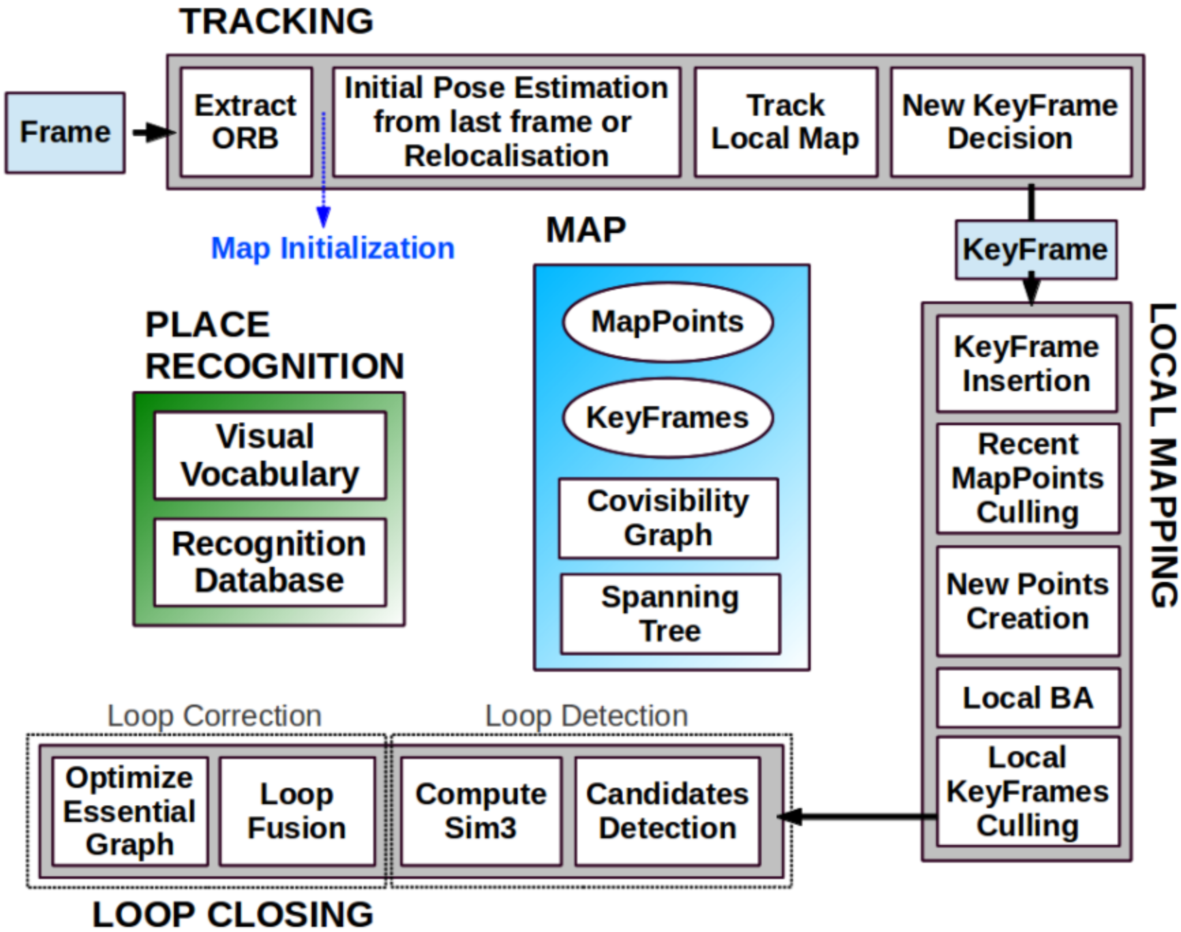
1）The first open source SLAM system for monocular, binocular and RGBD cameras, including loopback and relocation as well as map reuse.

2）The results of RGBD show that more accuracy can be obtained by using BA than ICP or minimization based on photometric and depth errors.

3) By using the far point and near point in the binoculars, as well as monocular observation, the binocular results are more accurate than the direct binocular SLAM algorithm.

4) Light positioning mode can effectively reuse maps.

ORB-SLAM2 includes modules common to all SLAM systems: tracking, mapping, relocalization, and loop closing. The figure below is the process of ORB-SLAM2.



8.2.Official case

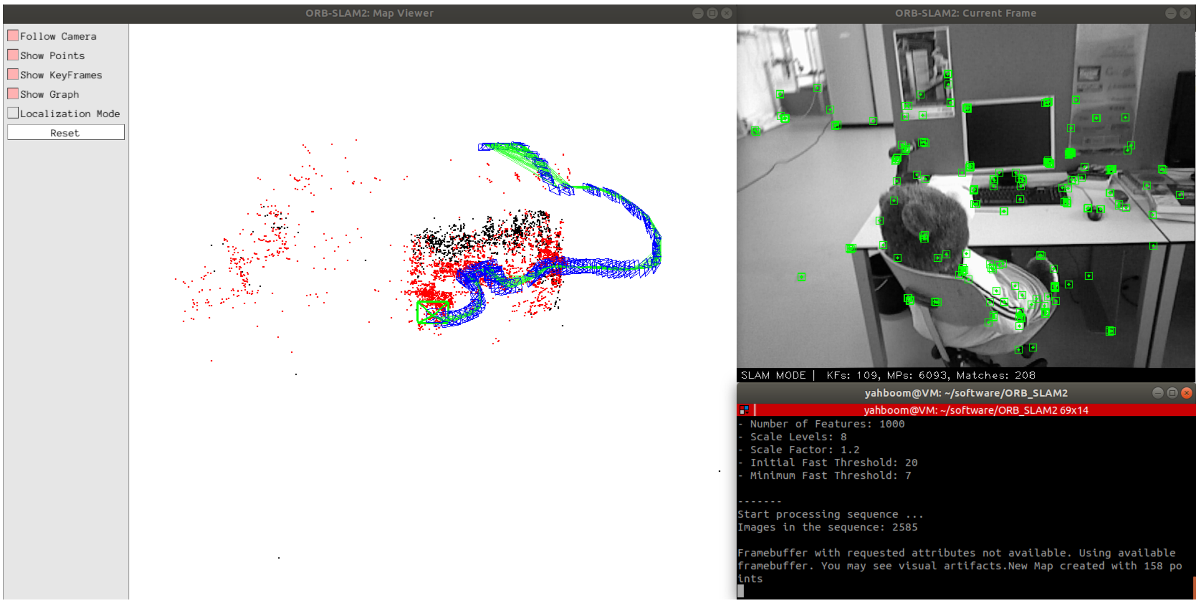
Open the terminal and enter ORB_SLAM2

```
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to Jetson Orin-Docker/05, Enter the robot's docker container
~/run_docker.sh
#Multiple ros commands require multiple terminals to enter the same docker container for execution, please refer to Jetson Orin-Docker/05, Section 5.8 tutorial
```

```
cd ~/software/ORB_SLAM2
```

8.2.1. Monocular test

```
./Examples/Monocular/mono_tum Vocabulary/ORBvoc.txt Examples/Monocular/TUM3.yaml ~/software/ORB_SLAM_data/rgbd_dataset_freiburg3_long_office_household
```



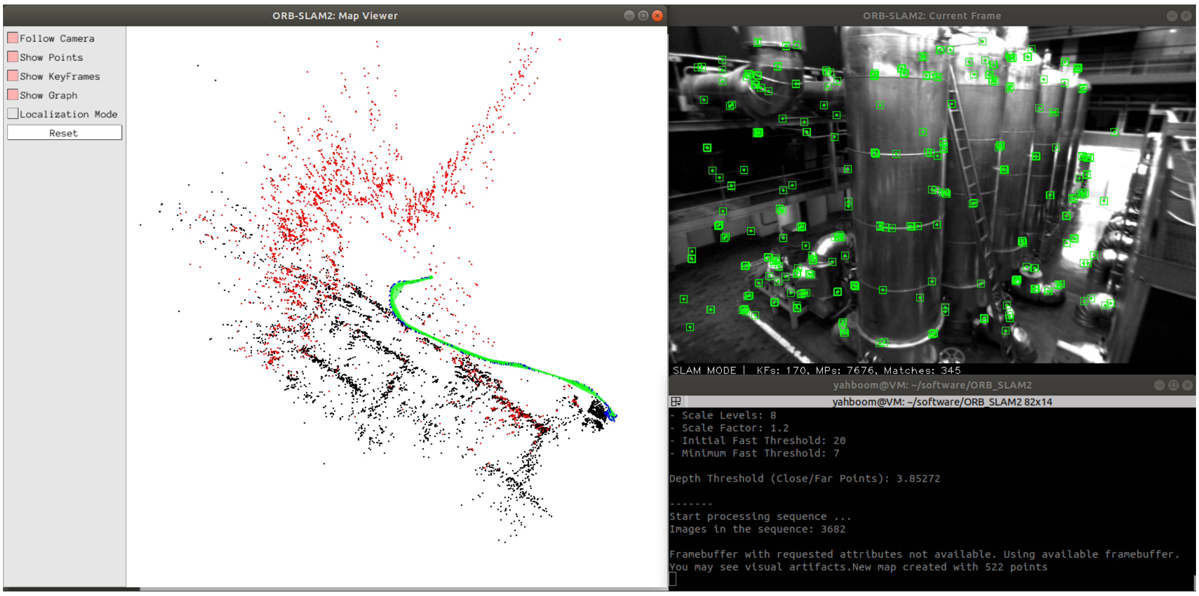
The blue frame is the key frame, the green frame is the camera orientation, the black point is the saved point, and the red point is the point currently seen by the camera.

After the test is completed, the keyframes are saved to the KeyFrameTrajectory.txt file in the current directory.

```
# Timestamp position (x y z) +attitude (x y z w)
1341847980.722988 -0.0000464 0.0001060 0.0000110 -0.0000183 0.0001468 -0.0000286
1.0000000
```

8.2.2.Binocular test

```
./Examples/Stereo/stereo_euroc Vocabulary/ORBvoc.txt Examples/Stereo/EuRoC.yaml
~/software/ORB_SLAM_data/MH_01_easy/mav0/cam0/data
~/software/ORB_SLAM_data/MH_01_easy/mav0/cam1/data
Examples/Stereo/EuRoC_TimeStamps/MH01.txt
```



The blue frame is the key frame, the green frame is the camera orientation, the black point is the saved point, and the red point is the point currently seen by the camera.

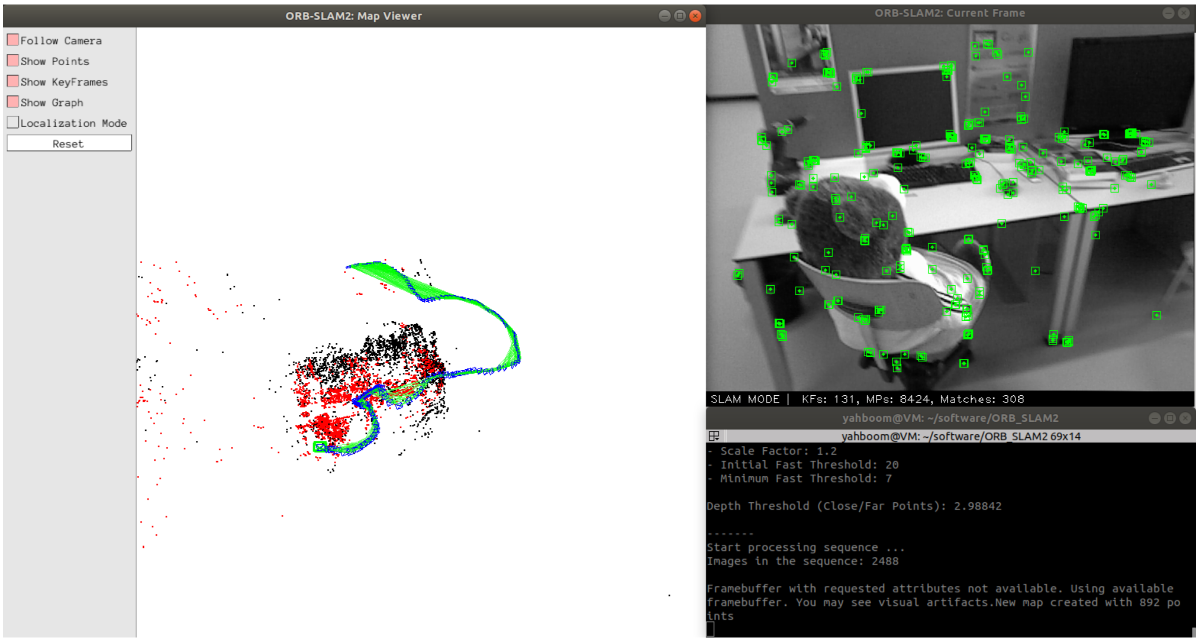
After the test is completed, the keyframes are saved to the CameraTrajectory.txt file in the current directory.

```
#Timestamp position (x y z) +attitude (x y z w)
1403636597.963556 -0.020445164 0.127641633 0.107868195 -0.136788622 -0.074876986
-0.044620439 0.986757994
```

8.2.3.RGBD test

test command

```
cd ~/software/ORB_SLAM_data
./Examples/RGB-D/rgbd_tum Vocabulary/ORBvoc.txt Examples/RGB-D/TUM3.yaml
~/software/ORB_SLAM_data/rgbd_dataset_freiburg3_long_office_household
~/software/ORB_SLAM_data/rgbd_dataset_freiburg3_long_office_household/association
s.txt
```



8.3. ORB_SLAM2_ROS camera test

The camera's internal parameters have been modified before the product leaves the factory. If you want to learn how to do this, please refer to the section [8.3.1, Internal Parameter Modification]. It can be handheld or used as a robot as a mobile carrier for mobile testing.

If it is held, there is no need to execute the next command, otherwise, it will be executed. (Robot terminal)

```
roslaunch yahboomcar_slam camera_driver.launch
```

Start camera ORB_SLAM2 testing (Robot side)

```
roslaunch yahboomcar_slam camera_orb_slam.launch orb_slam_type:=mono
```

- 【orb_slam_type】parameter：[mono,monoAR,rgbd]，There are three types available, monocular, monocular AR, and rgbd.

8.3.1. Modification of internal parameters

The camera requires the internal parameters of the camera before running ORBSLAM, so the camera must be calibrated first. The specific method can be found in the lesson [02, Astra Camera Calibration].

Start monocular camera

```
roslaunch usb_cam usb_cam-test.launch
```

Start calibration node

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9    "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$
```

2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9    "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$ docker exec -it 5b698ea10535 /bin/bash
---------------------------------------------------------
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
---------------------------------------------------------
root@ubuntu:/#
```

After successfully entering the container, you can open countless terminals to enter the container.

```
rosrun camera_calibration cameracalibrator.py image:=/usb_cam/image_raw
camera:=/usb_cam --size 9x6 --square 0.02
```

After the calibration is completed, move the [calibrationdata.tar.gz] file to the [home] directory.

```
sudo mv /tmp/calibrationdata.tar.gz ~
```

After unzipping, open [ost.yaml] in the folder and find the camera internal parameter matrix, for example: the following content.

```
camera_matrix:
  rows: 3
  cols: 3
  data: [ 683.90304,    0.     ,  294.56102,
            0.     ,  679.88513,  228.05956,
            0.     ,    0.     ,    1.     ]
```

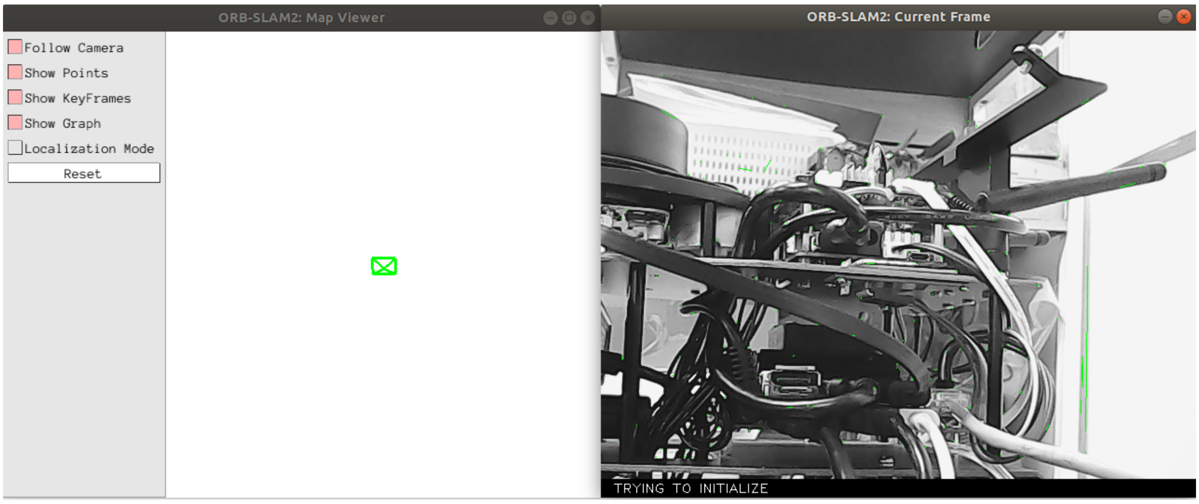
Camera internal parameter matrix

```
# fx 0 cx
# 0 fy cy
# 0 0  1
```

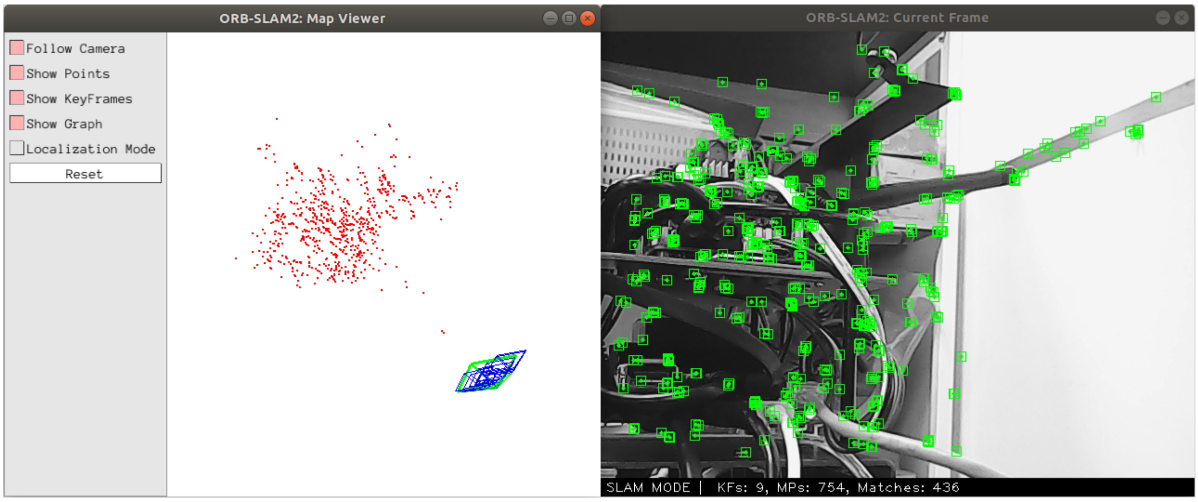
Modify the data in data to the values corresponding to [astra.yaml] and [astra1.0.yaml] in the [param] folder under the [yahboomcar_slam] function package.

```
Camera.fx: 683.90304
Camera.fy: 679.88513
Camera.cx: 294.56102
Camera.cy: 228.05956
```

8.3.2. Monocular

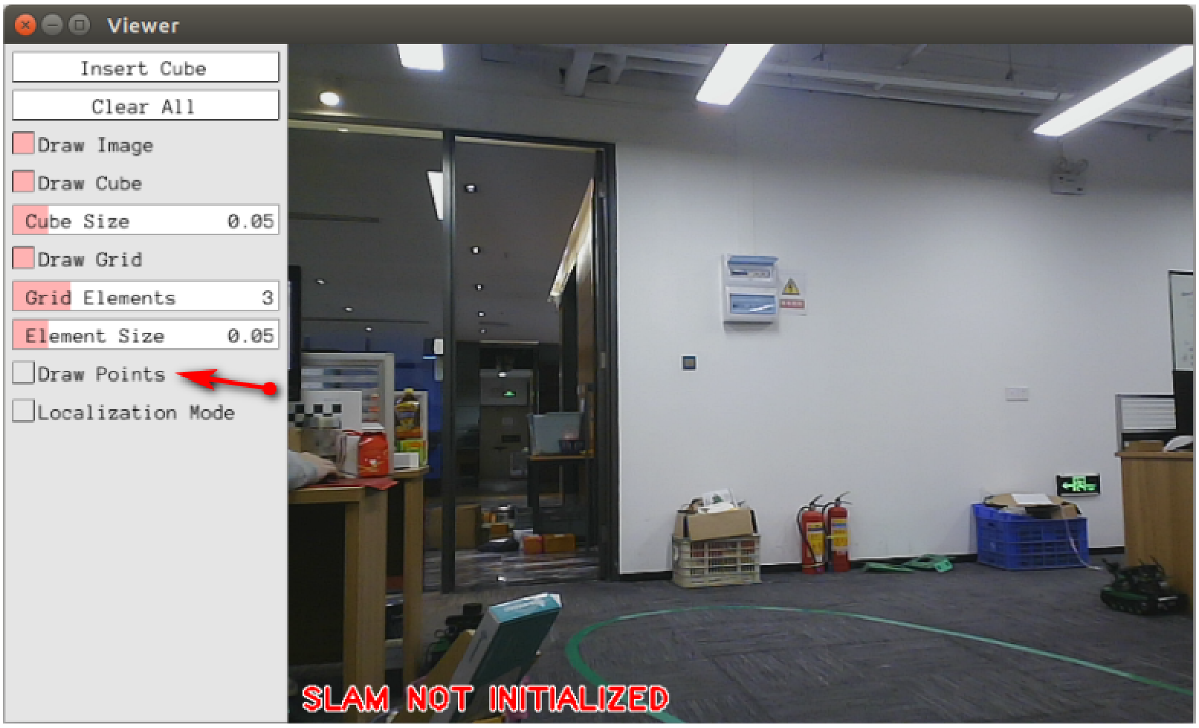


When the command is executed, there is only a green box in the [ORB_SLAM2:Map Viewer] interface, and the [ORB_SLAM2:Current Frame] interface is trying to initialize. At this time, slowly move the camera up, down, left, and right to find feature points in the screen and initialize slam. .

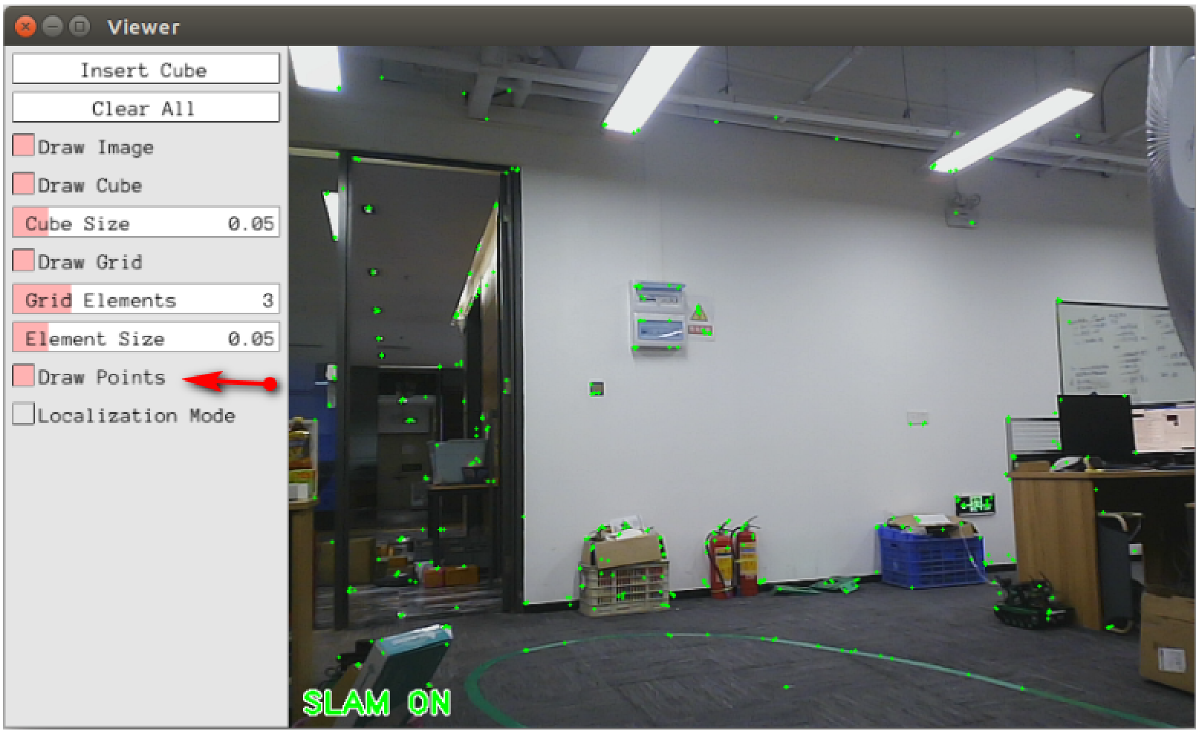


As shown in the picture above, when you enter the [SLAM MODE] mode, you must continuously obtain each frame of image to position the camera when running the monocular. If you select the pure positioning mode of [Localization Mode] in the upper left picture, the camera will not be able to find its position. You have to start over to get the keyframes.

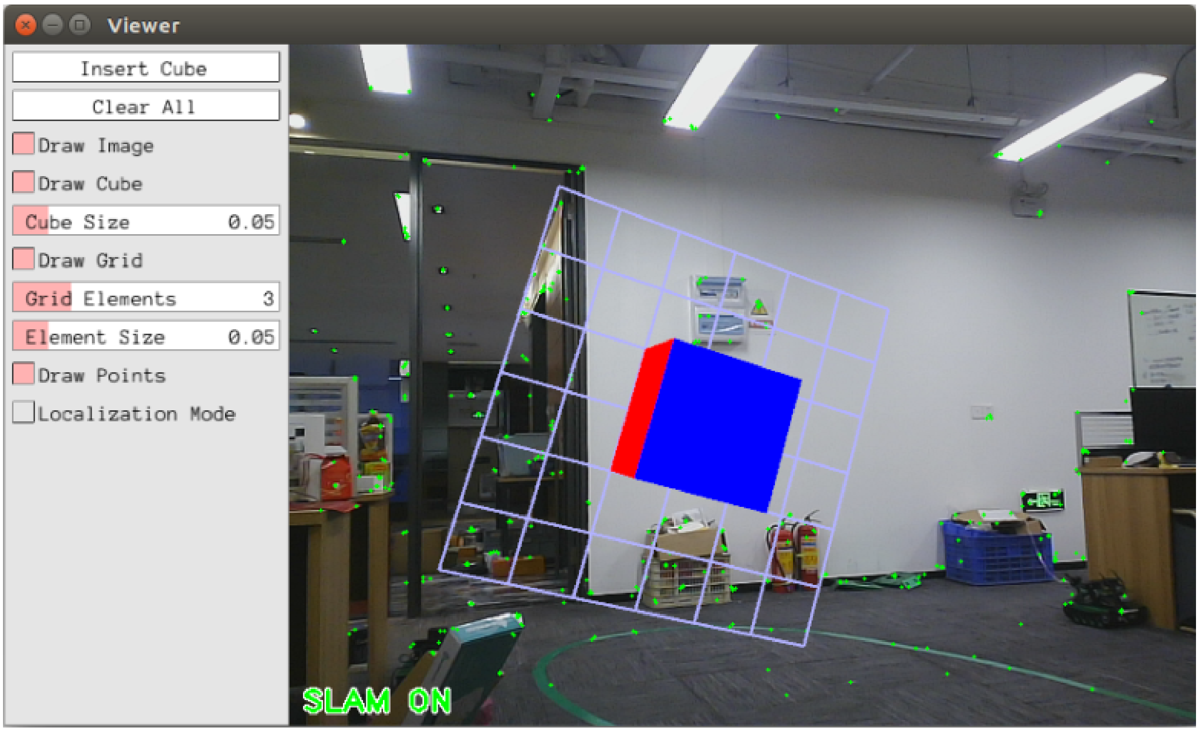
8.3.3.Monocular AR



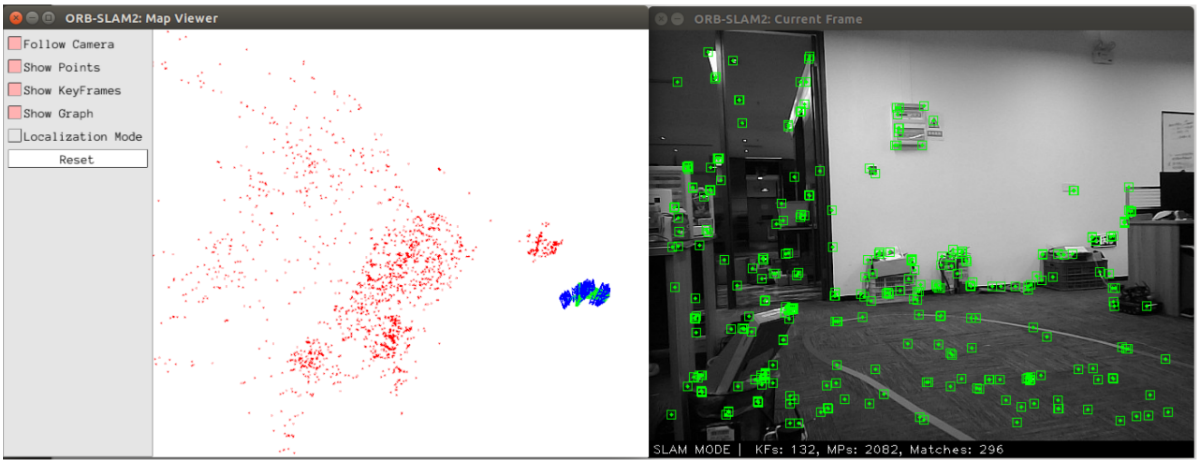
When the command is executed, there is only one interface and [slam not initialized] is displayed. slam is not initialized. Click the box to the left of [Draw Points] in the left column to display feature points. At this time, slowly move the camera up, down, left, and right to find feature points in the picture and initialize slam.



As shown in the picture above, enter [SLAM ON] mode at this time. Click [Insert Cube] on the screen to insert an AR cube where it is considered to be a plane. And the AR block will always be in a fixed position in the scene.



8.3.4. RGBD



RGBD does not have to continuously acquire each frame of image like running a monocular. If you select the pure positioning mode [Localization Mode] in the upper left picture, you can position the key frame just acquired.