5.ORB_SLAM2 basics

1.Brief introduction

ORB-SLAM is mainly used for monocular SLAM;
The ORB-SLAM2 version supports three interfaces: monocular, binocular, and RGBD;
The ORB-SLAM3 version adds IMU coupling and supports fisheye cameras.
All steps of ORB-SLAM uniformly use the ORB features of the image. ORB feature is a very fast feature extraction method that has rotation invariance and can be used to construct scale invariance using a pyramid. The use of unified ORB features helps SLAM algorithm achieve consistency in feature extraction and tracking, keyframe selection, 3D reconstruction, closed-loop detection, and other steps. The system is also robust to intense motion and supports wide baseline closed-loop detection and relocation, including fully automatic initialization. Due to the fact that the ORB-SLAM system is based on feature points, it can calculate the trajectory of the camera in real-time and generate sparse 3D reconstruction results of the scene.
On the basis of ORB-SLAM, ORB-SLAM2 contribution points:

1. The first open-source SLAM system for monocular, binocular, and RGBD cameras, including loopback and repositioning, as well as map reuse.
2. The results of RGBD show that using BA can achieve more accurate results than ICP or minimization based on photometric and depth errors.
3. By utilizing the far and near points in binoculars, as well as monocular observations, the binocular results are more accurate than the direct binocular SLAM algorithm.
4. The light positioning mode can effectively reuse maps.
   ORB-SLAM2 includes modules common to all SLAM systems: Tracking, Mapping, Relocation, and Loop Closing.. The following diagram shows the process of ORB-SLAM2.

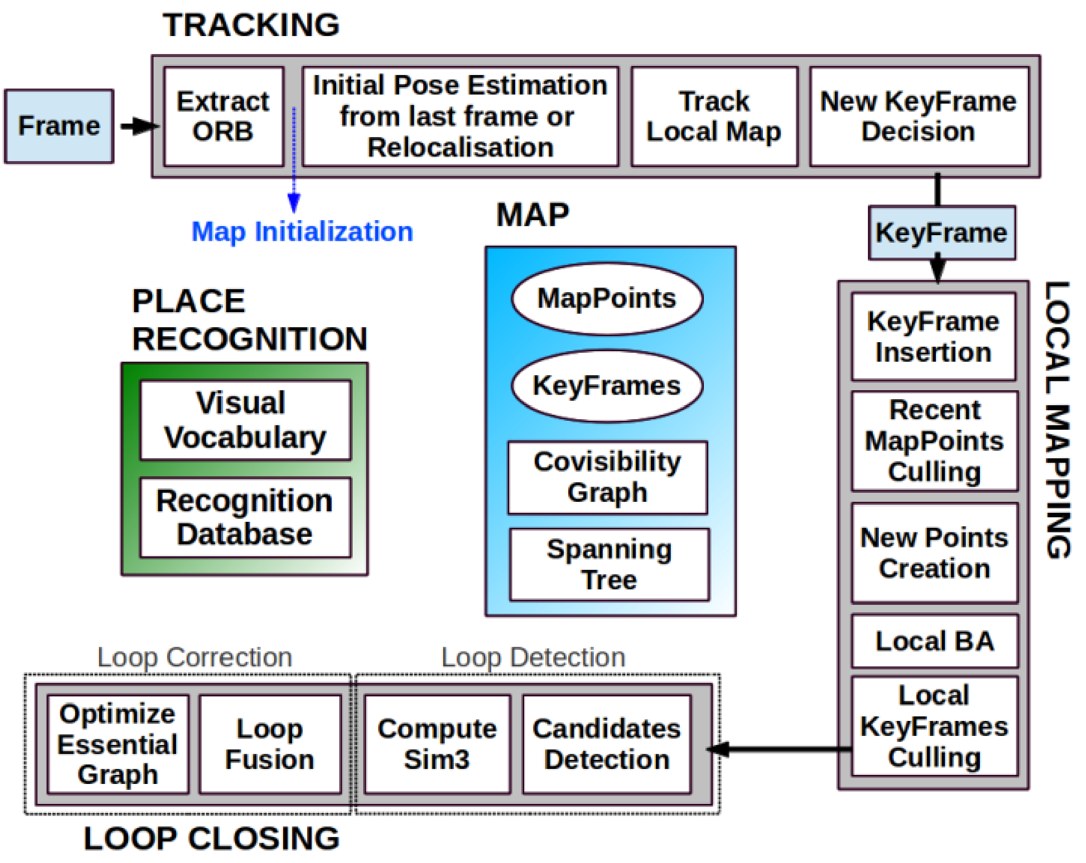


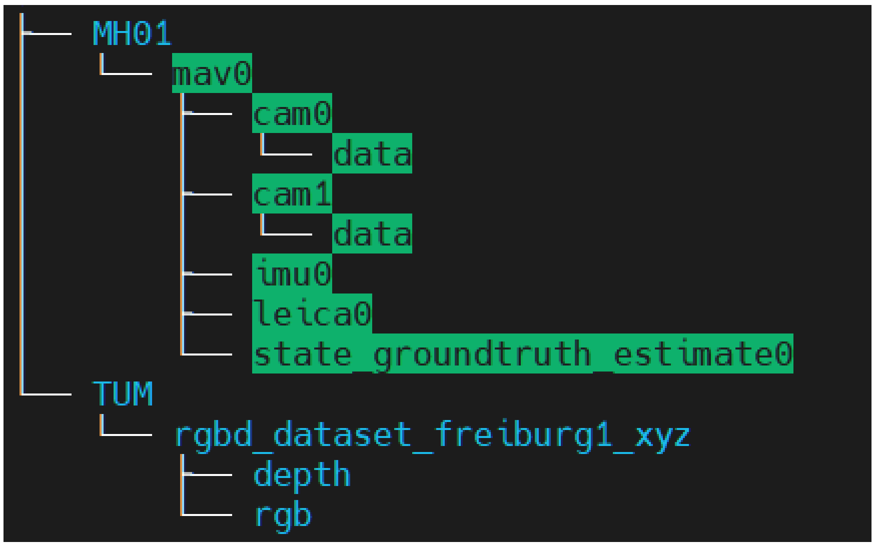
Fig. 1. ORB-SLAM system overview, showing all the steps performed by the tracking, local mapping and loop closing threads. The main components of the place recognition module and the map are also shown.

2. Case

2.1 Dataset location

```
/root/yahboomcar_ros2_ws/software/orbslam2/ORB_SLAM2-master/data
```

Here we downloaded the EuRoC MH01 dataset and TUM rgbd_ Dataset_ Freiburg1_ Xyz dataset.



```
├── MH01
│   └── mav0
│       ├── cam0
│       │   └── data
│       ├── cam1
│       │   └── data
│       ├── imu0
│       ├── leica0
│       └── state_groundtruth_estimate0
└── TUM
    └── rgbd_dataset_freiburg1_xyz
        ├── depth
        └── rgb
```

If you need other datasets, you can download them at the following address:

```
TUM Dataset: http://vision.in.tum.de/data/datasets/rgbd-dataset/download
KITTI Dataset: http://www.cvlibs.net/datasets/kitti/eval_odometry.php
EuRoC Dataset: http://projects.asl.ethz.ch/datasets/doku.php?
id=kmavvisualinertialdatasets
```

2.2 Monocular testing

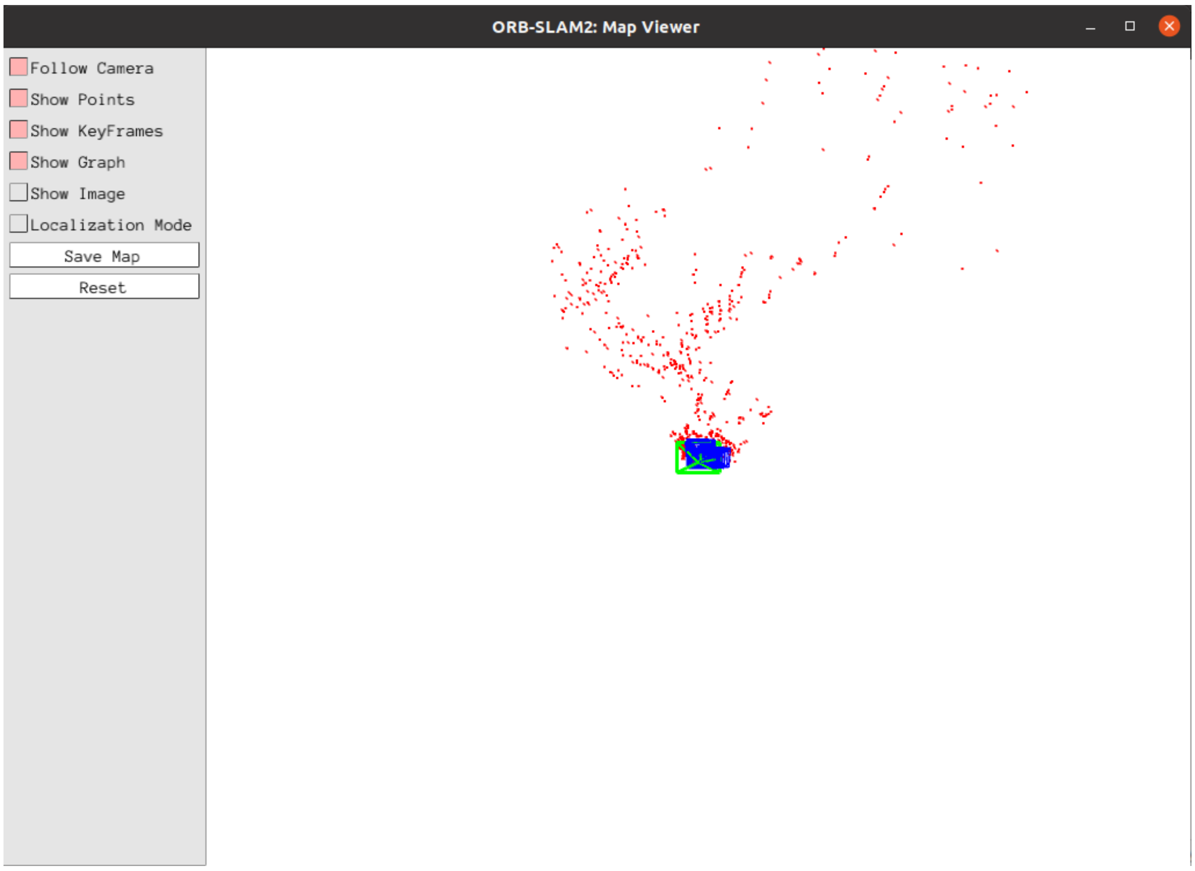
Taking the EuRoC dataset as an example, enter ORB_ SLAM2 directory:

```
cd /home/yahboom/orbslam2/ORB_SLAM2
```

Input following command:

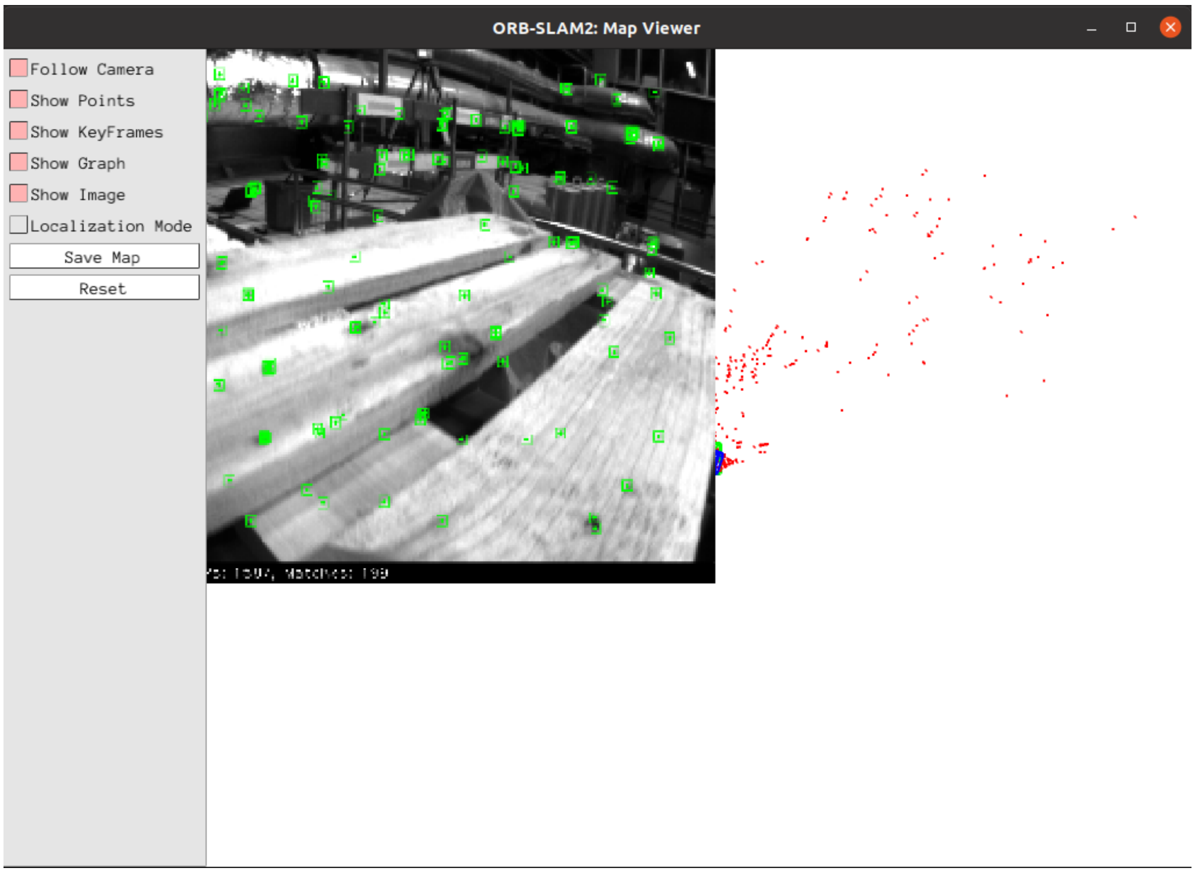
```
Examples/Monocular/mono_euroc Vocabulary/ORBvoc.txt Examples/Monocular/EuRoC.yaml
data/MH_01_easy/mav0/cam0/data Examples/Monocular/EuRoC_TimeStamps/MH01.txt
```

After successful operation, the interface is shown in the following figure.



The blue box represents the keyframes, the green box represents the camera orientation, the black dots represent the saved points, and the red dots represent the points currently seen by the camera.

Click on 【Show Image】 to view the image.



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

```
#Time stamp 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
```

2.3 Binocular testing

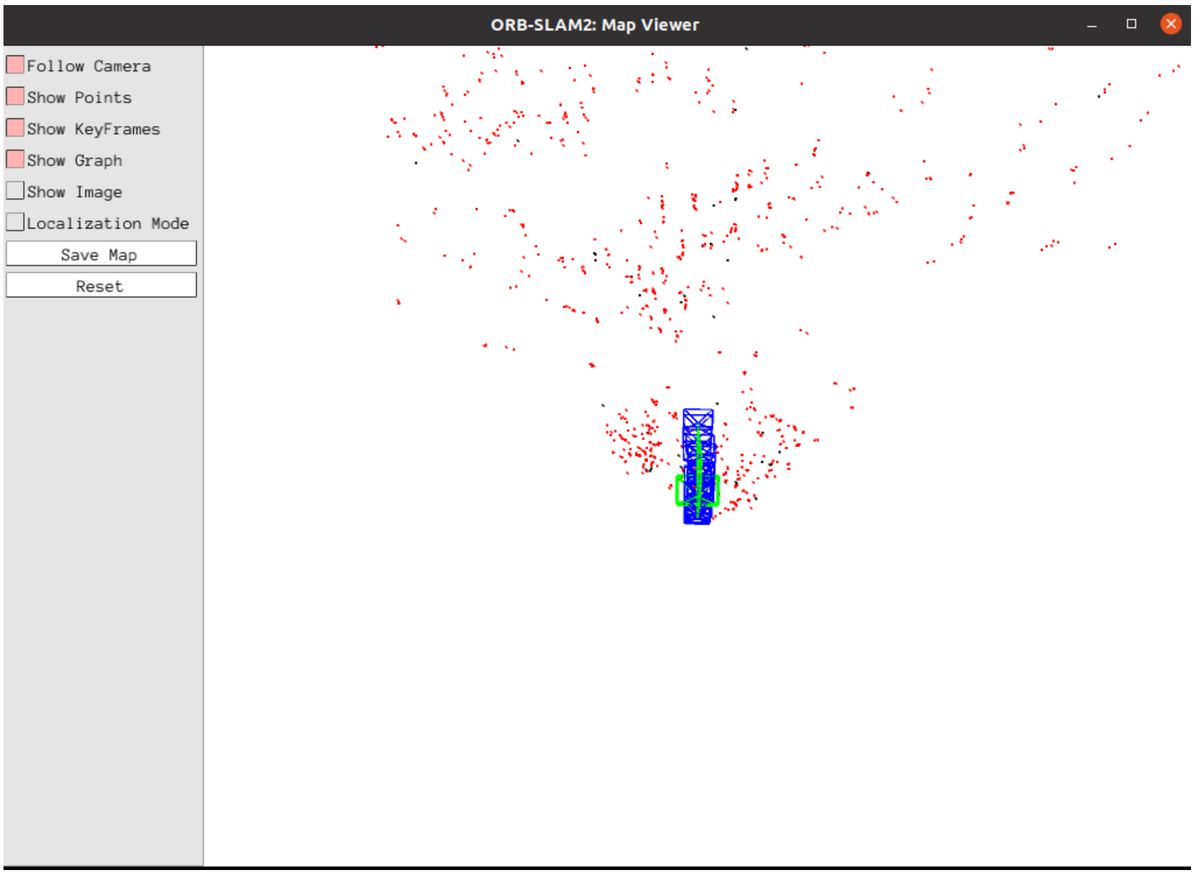
Taking the EuRoC dataset as an example, enter ORB_ Run the following command in the SLAM2 directory.

```
cd /home/yahboom/orbslam2/ORB_SLAM2
```

Input following command:

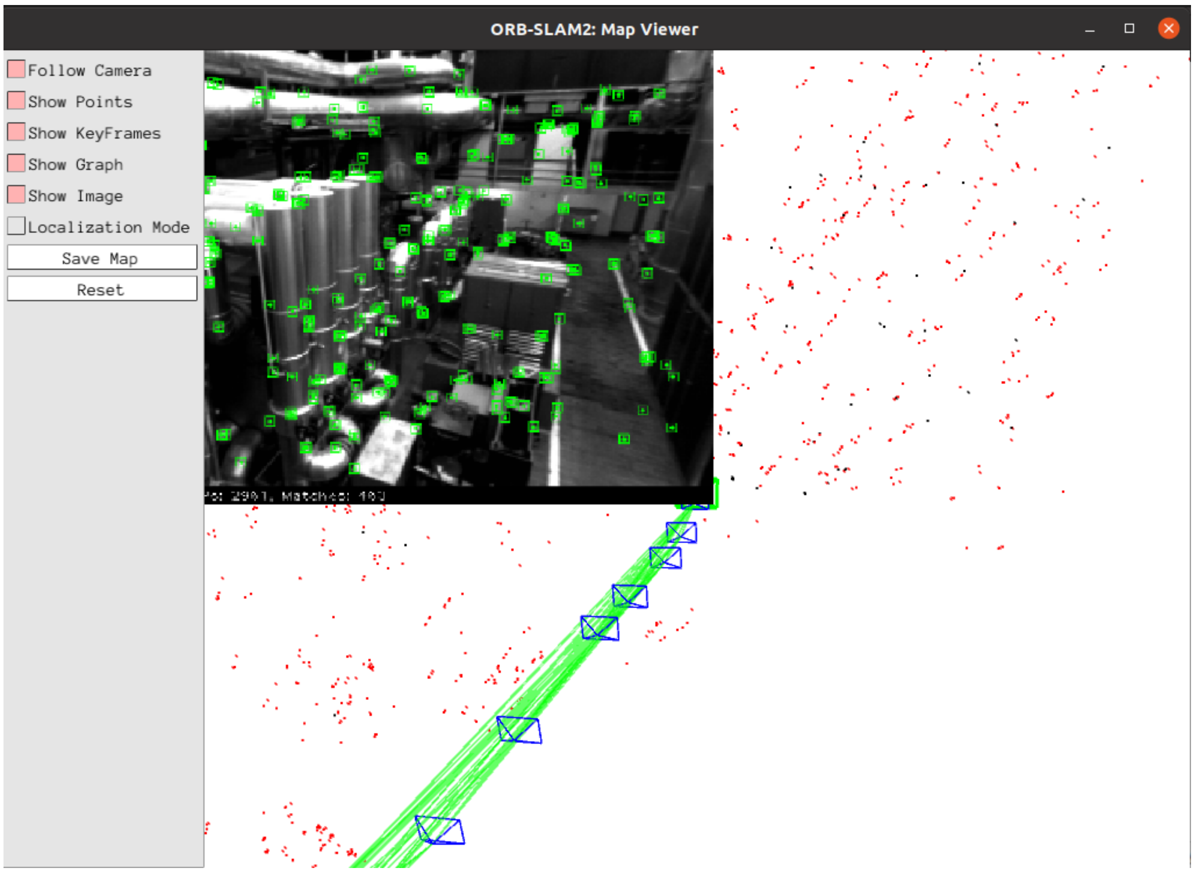
```
Examples/Stereo/stereo_euroc Vocabulary/ORBvoc.txt Examples/Stereo/EuRoC.yaml
data/MH_01_easy/mav0/cam0/data data/MH_01_easy/mav0/cam1/data
Examples/Stereo/EuRoC_TimeStamps/MH01.txt
```

After successful operation, the interface is shown in the following figure.



The blue box represents the keyframe, the green box represents the camera orientation, the black point represents the saved point, and the red point represents the point currently seen by the camera.

Click on 【Show Image】 to view the image.



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

```
#Time stamp 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
```

2.4 RGBD testing

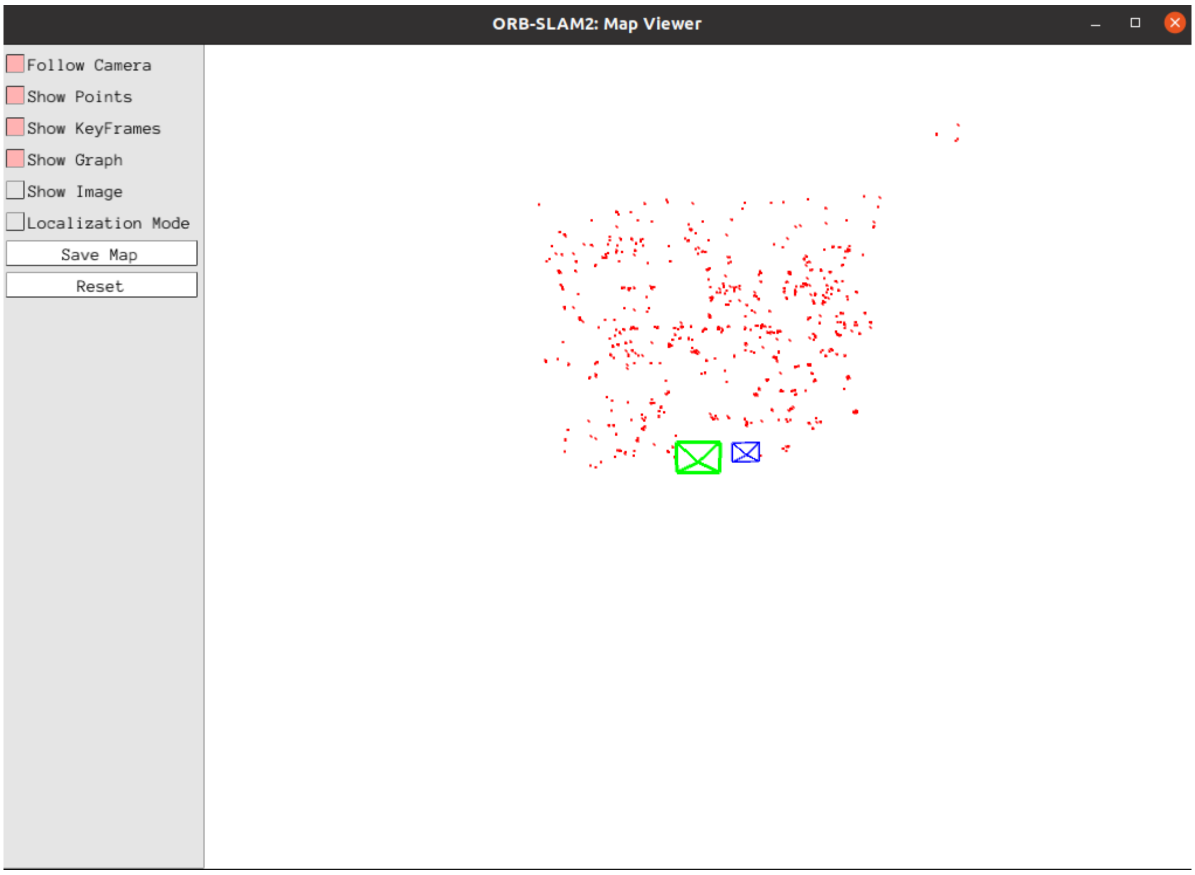
Enter ORB_ In the SLAM2 directory, input following command

```
cd /home/yahboom/orbslam2/ORB_SLAM2
```

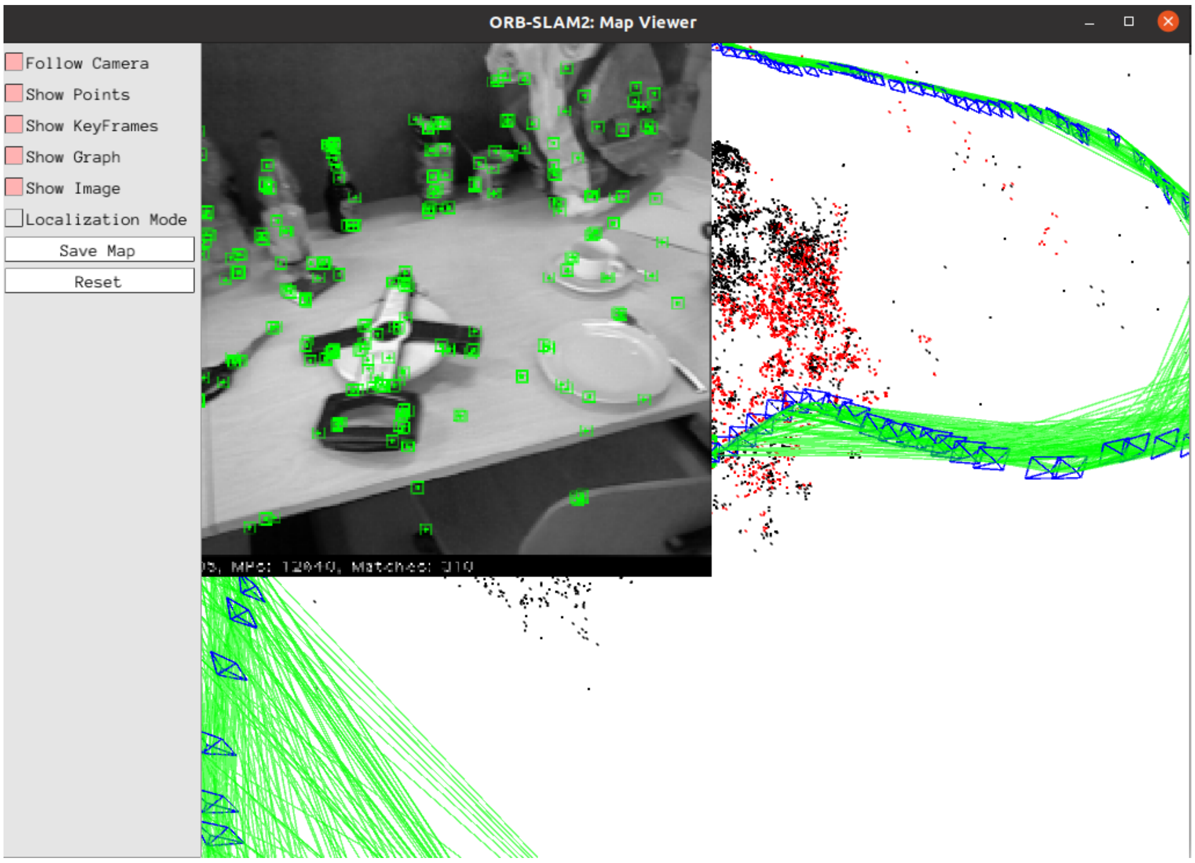
Input following command:

```
Examples/RGB-D/rgbd_tum Vocabulary/ORBvoc.txt Examples/RGB-D/TUM1.yaml
data/rgbd_dataset_freiburg3_long_office_household
data/rgbd_dataset_freiburg3_long_office_household/associations.txt
```

After successful operation, the interface is shown in the following figure.



Click on 【Show Image】 to view the image.



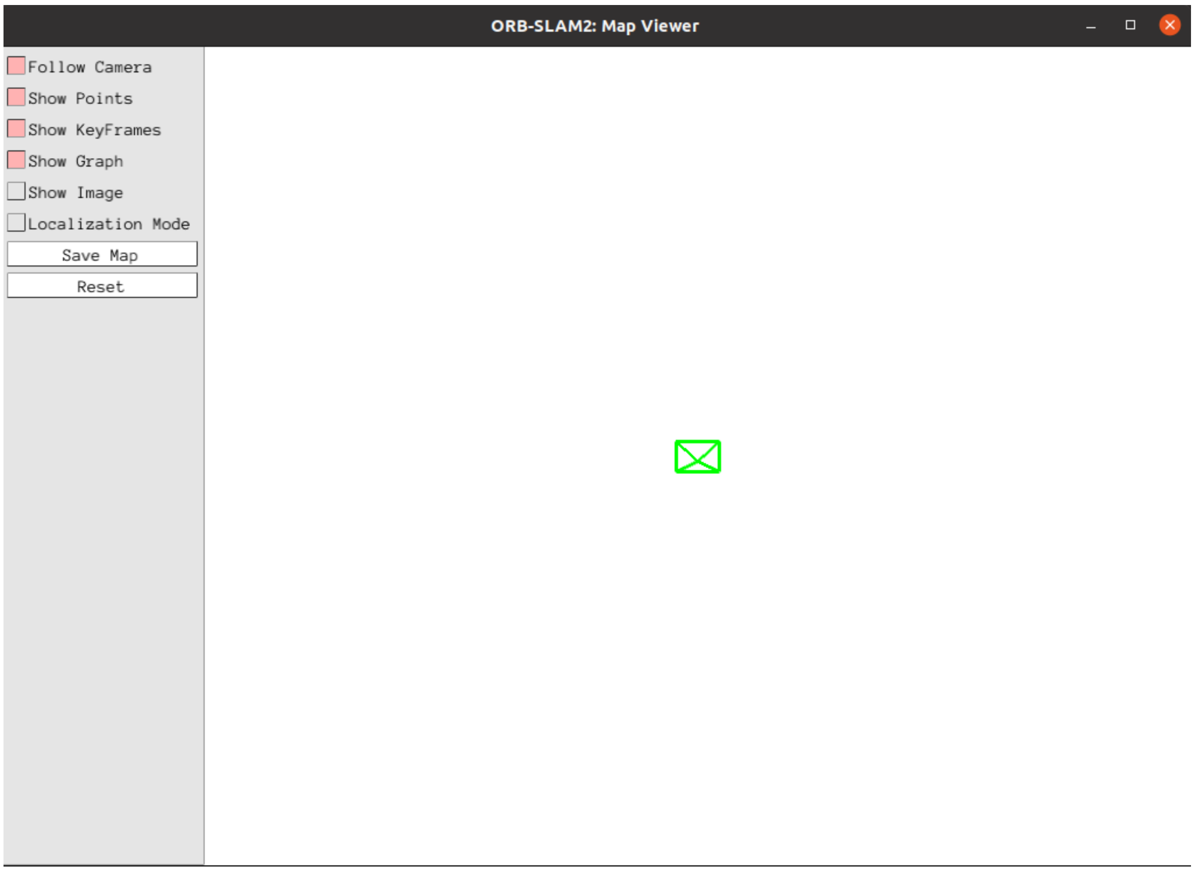
After running, CameraTrajectory.txt and KeyFrameTrajectory.txt will also be saved

3. ORB_SLAM2 ROS2 camera test

3.1 Monocular testing

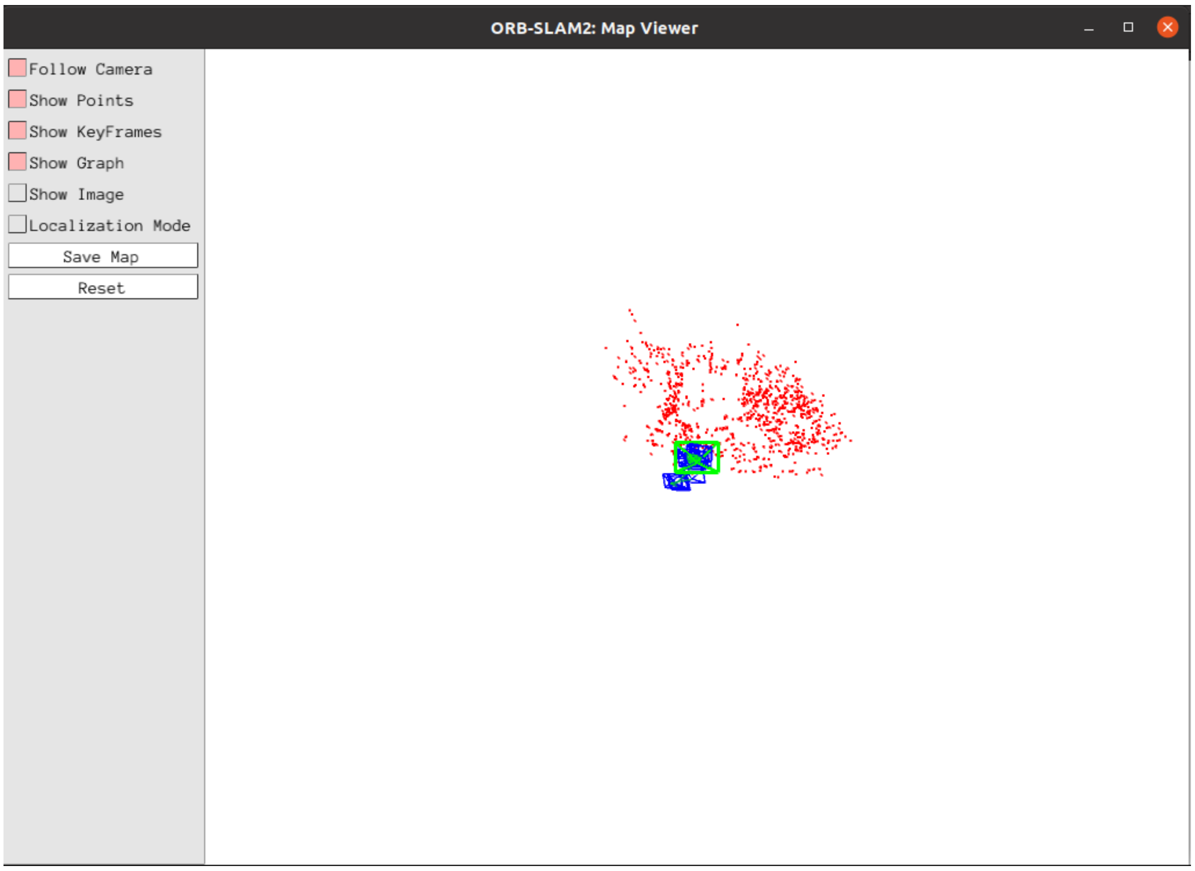
Start camera ORB_ SLAM2 test

```
# Activate camera: only dabai is used here_ Rgb image of dcw2 camera (equivalent to monocular)
ros2 launch orbbec_camera dabai_dcw2.launch.py
# Start the orbslam node
ros2 launch yahboomcar_slam orbslam_ros_launch.py orb_slam_type:=mono
```

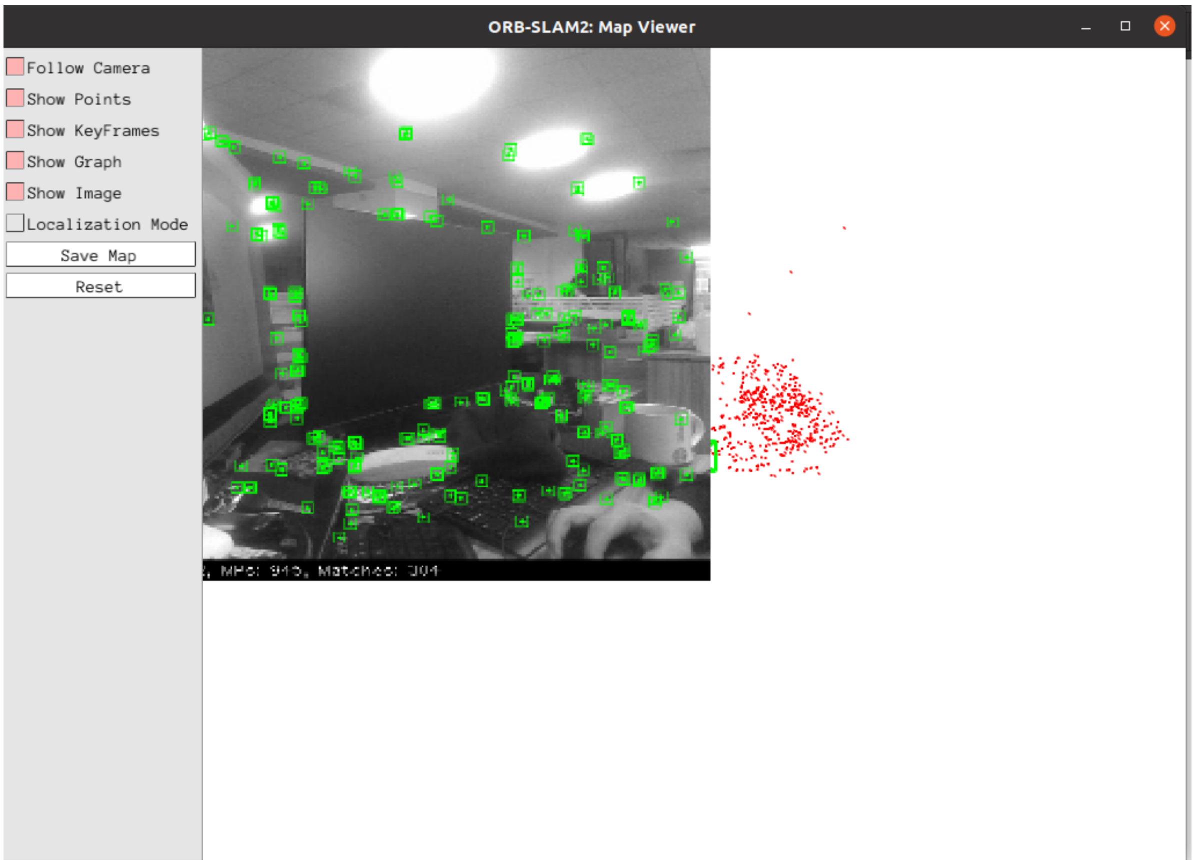


When the command is executed, there is only a green box in the interface of the ORB_SLAM2: Map Viewer. The initialization is being attempted in the interface of the ORB_SLAM2: Current Frame.

At this time, slowly move the camera up, down, left, right, and search for feature points in the screen and initialize the slam.

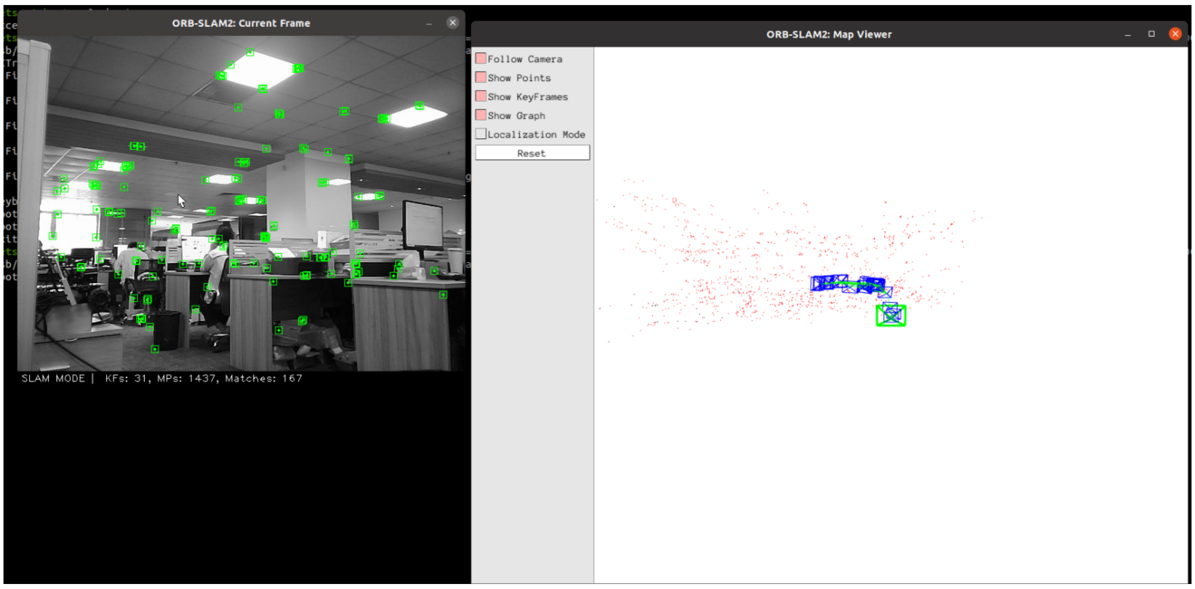


Click on 【Show Image】 to view the image.



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

```
~/orbbec_ws/src/ros2-ORB_SLAM2/src//monocular/KeyFrameTrajectory.txt
```



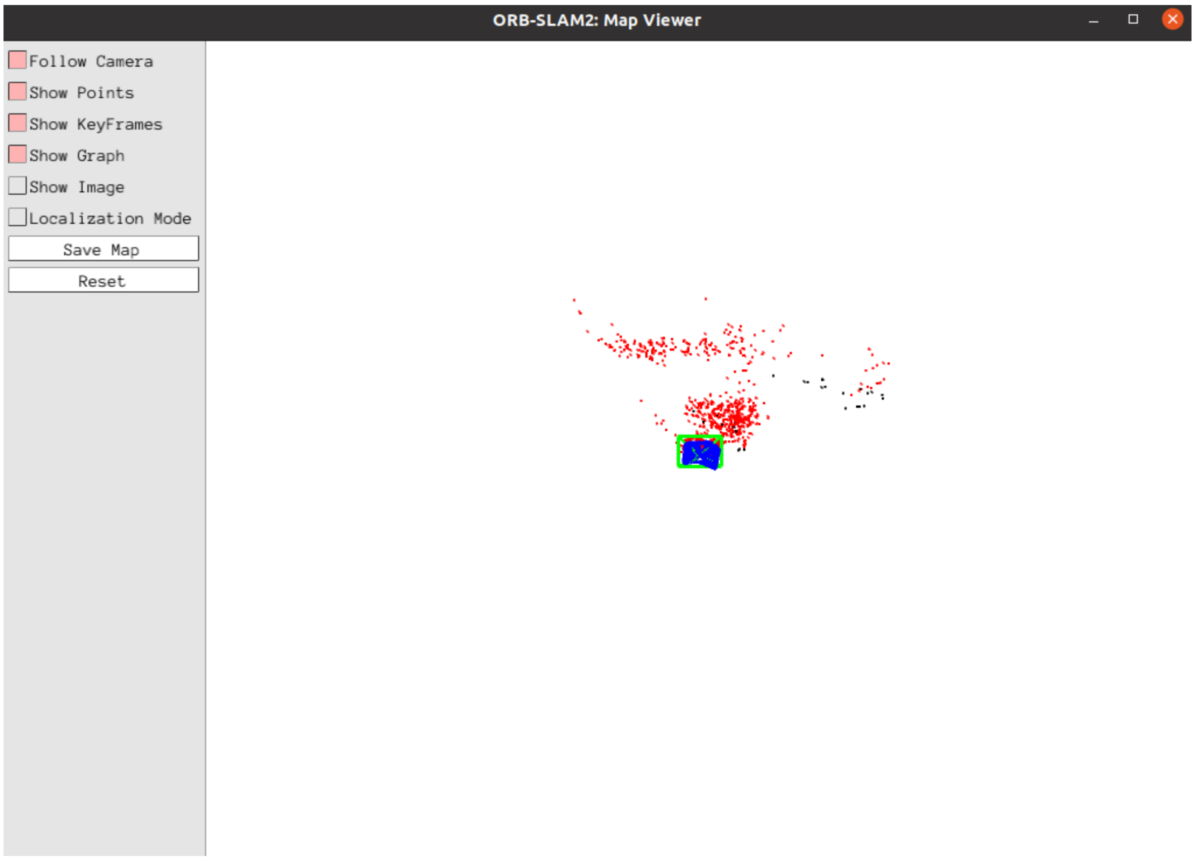
As shown in the above figure, at this point, the camera enters the [SLAM Mode] mode and must continuously obtain each frame of image for camera positioning. If the [Localization Mode] pure positioning mode in the left image is selected, the camera will not be able to find its own position and must start from scratch to obtain key frames.

3.2 RGBD test

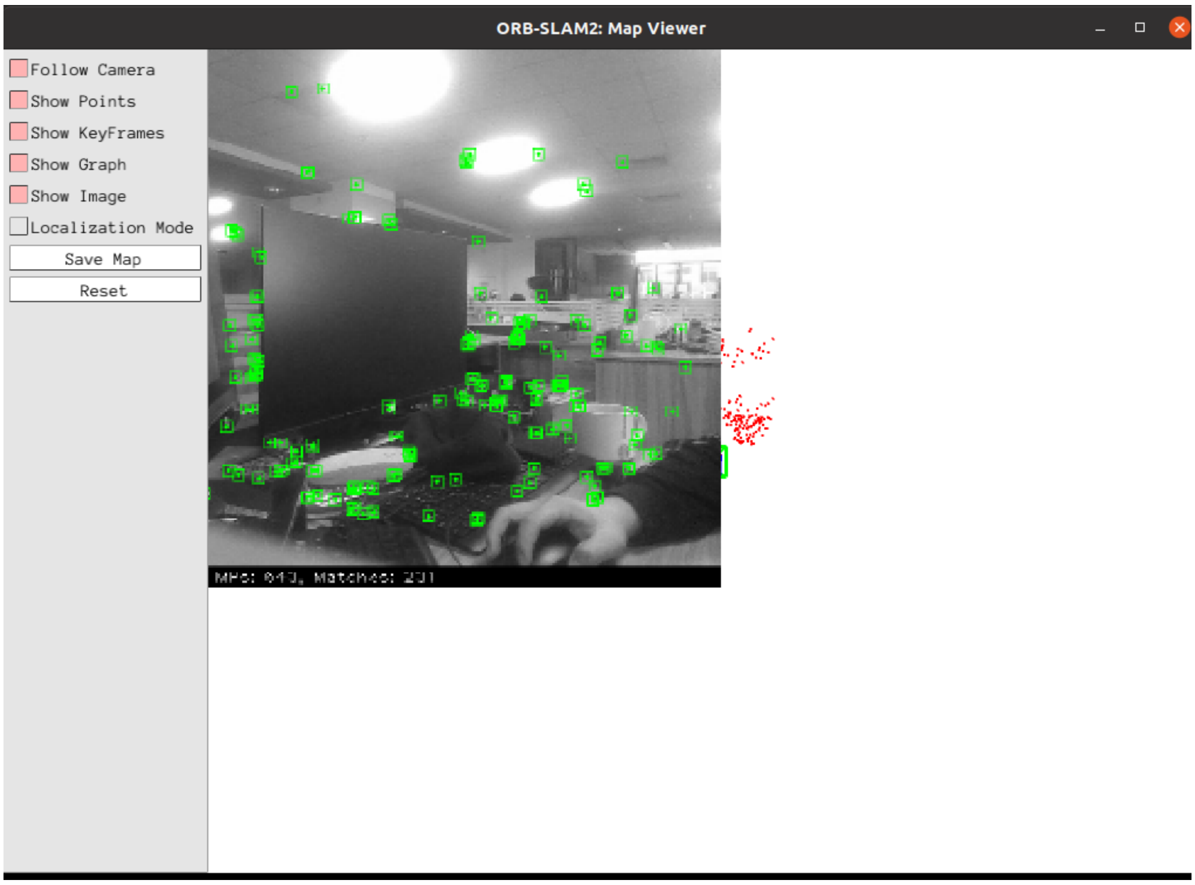
```
# Start camera
ros2 launch orbbec_camera dabai_dcw2.launch.py
# Start the orbslam node
ros2 launch yahboomcar_slam orbslam_ros_launch.py orb_slam_type:=rgbd
```

RGBD does not need to continuously obtain each frame of image like running a single camera.

If you choose the pure localization mode in the upper left image, you can locate the key frame just obtained.



Click on 【Show Image】 to view the image.



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

```
~/orbbec_ws/src/ros2-ORB_SLAM2/src/rgbd/KeyFrameTrajectory.txt
```