9. ORB_SLAM2_Octomap

octomap official website: http://octomap.github.io/

octomap source code: https://github.com/OctoMap/octomap

octomap wiki: http://wiki.ros.org/octomap

octomap_server: http://wiki.ros.org/octomap_server

9.1. Introduction

Octomap uses the octree data structure to store the probabilistic occupancy map of the three-dimensional environment. OctoMap library implements a 3D occupancy grid mapping method, providing data structures and mapping algorithms in C++, which is particularly suitable for robots. The map implementation is based on octree.

It elegantly compresses and updates maps with adjustable resolution! It stores maps in the form of octotree (will be discussed later), which can save a lot of space compared to point clouds. The map created by octomap probably looks like this: (different resolutions from left to right)

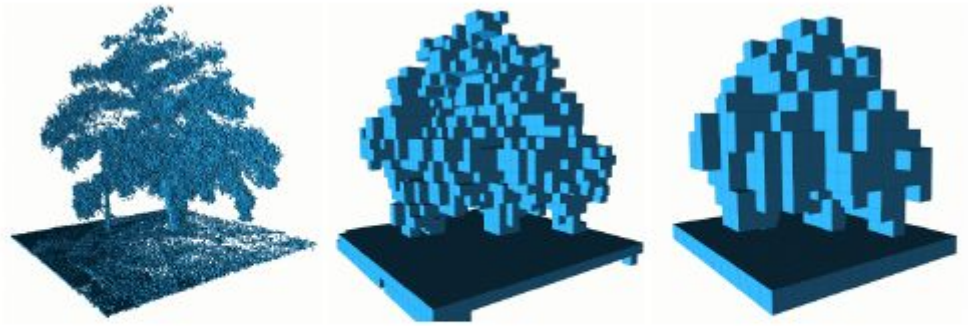


Fig. 3 By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

- Precautions

Note: When building a map, moving the robot slowly and losing key frames may cause the map building to fail.

According to different models, you only need to set the purchased model in [.bashrc], X1 (normal four-wheel drive) X3 (Mailun) Take X3 as an example

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameters and modify the corresponding car model

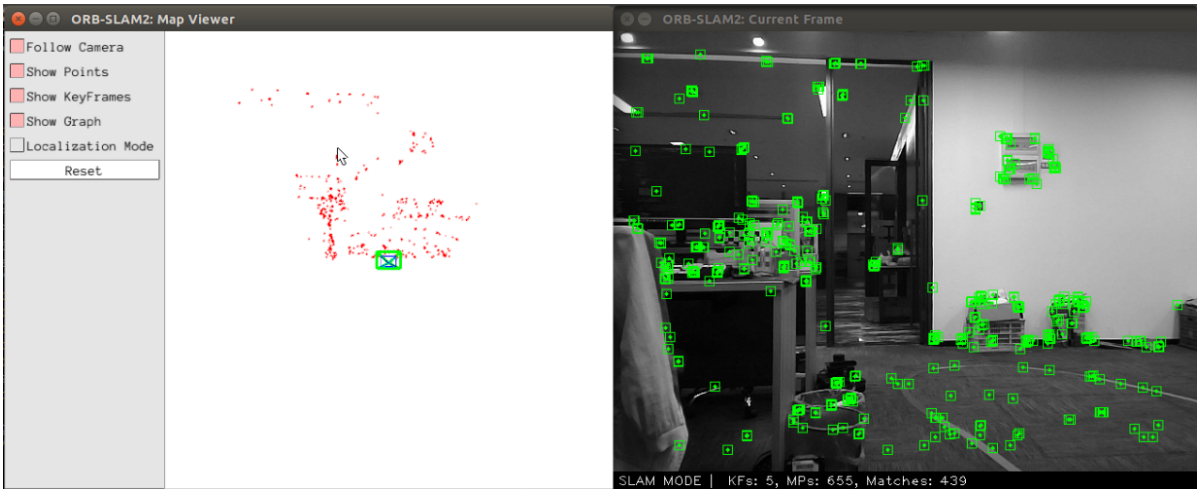
```
export ROBOT_TYPE=X3 # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

9.2. Use

Start orb_slam and the underlying driver (Robot side)

```
roslaunch yahboomcar_slam robot_orb_slam.launch bUseViewer:=true
```

- [bUseViewer] parameter: whether to open the visualization window of orbslam. If true, you can clearly view the key points. If the positioning is unsuccessful, you can reset the key points. Click [Reset] on the left side of the picture below.



<PI5 needs to open another terminal to enter the same docker container

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.

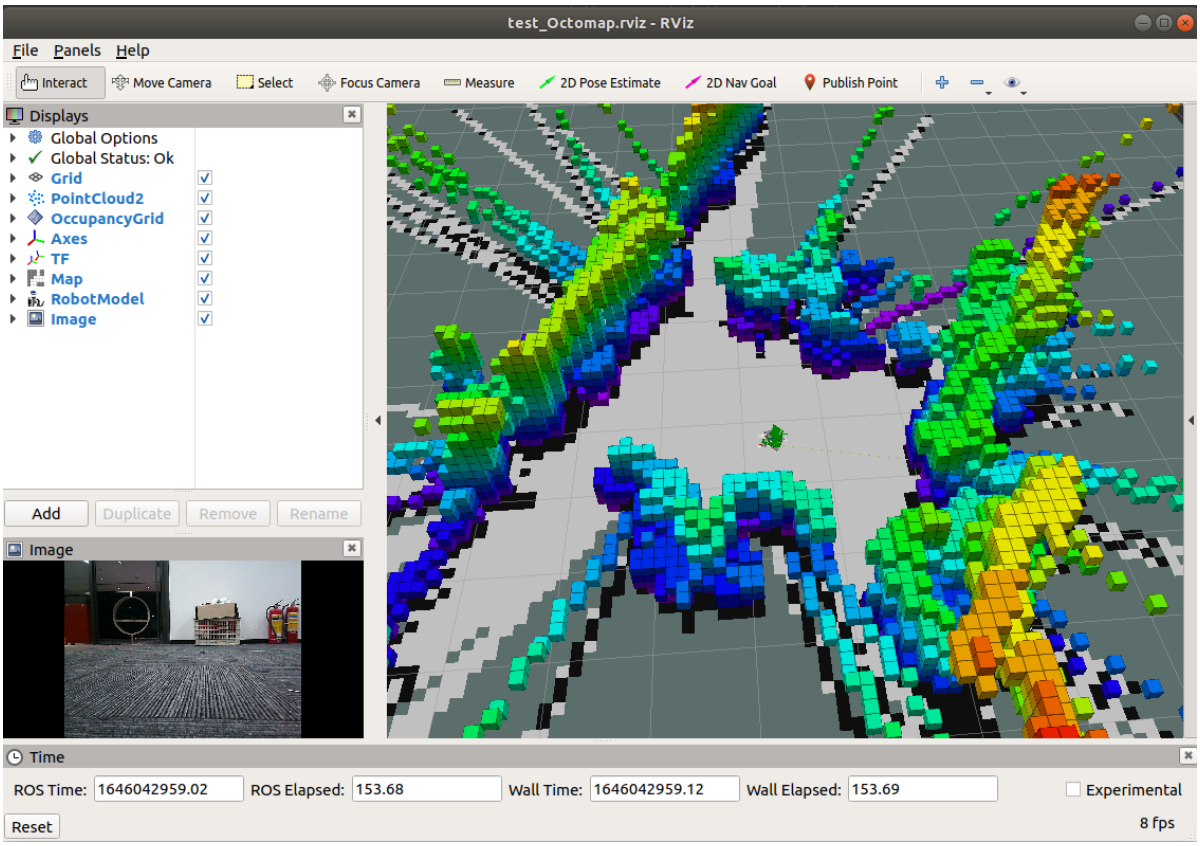
Start octree mapping (Robot side)

```
roslaunch yahboomcar_slam robot_orb_octomap.launch frame_id:=odom use_rviz:=false
```

- [frame_id] parameter: coordinate system name, available by default and no need to set.
- [use_rviz] parameter: whether to enable rviz.

Turn on the visual interface (virtual machine side)

```
roslaunch yahboomcar_slam view_orb_octomap.launch
```



Due to the octree, its map looks like it is composed of many small squares (much like Minecraft). When the resolution is high, the squares are small; when the resolution is low, the squares are large. Each square represents the probability of that square being occupied.

9.3, octomap_server

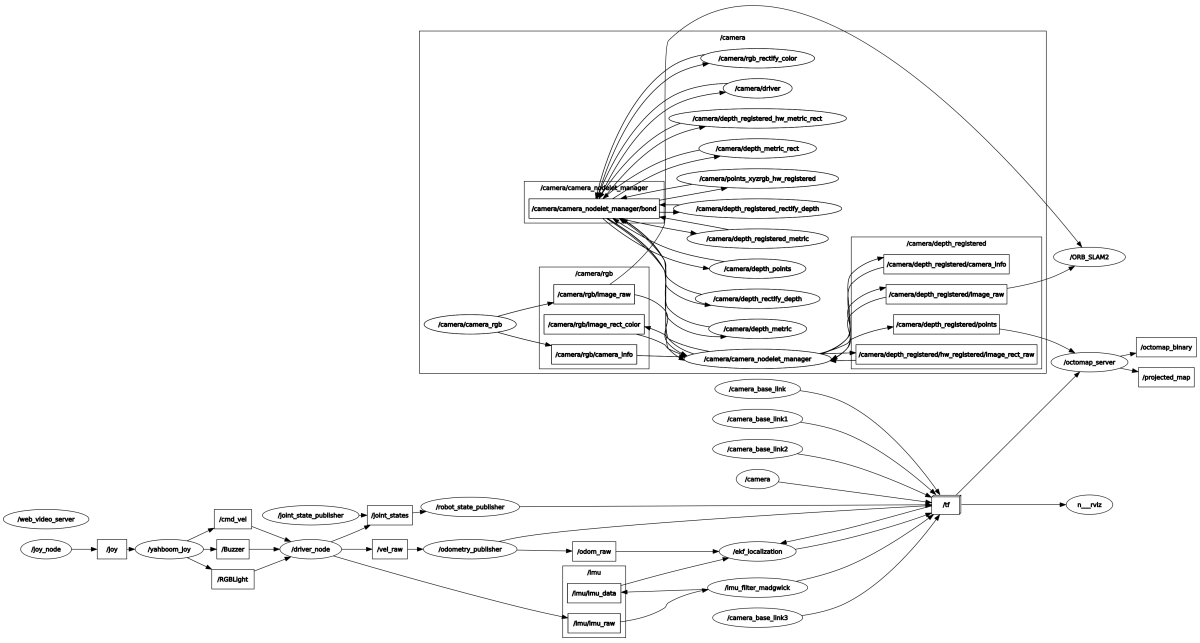
9.3.1, Topics and Services

Subscription topic	**Type**	**Description**
cloud_in	sensor_msgs/PointCloud2	Incoming 3D point cloud for scan integration.
Post Topic	**Type**	**Description**
octomap_binary	octomap_msgs/Octomap	Complete maximum likelihood occupancy map as a compact octal map binary stream, encoding free space and occupied space. Binary messages only differentiate between free space and occupied space, but smaller.
octomap_full	octomap_msgs/Octomap	Full maximum likelihood occupancy map as a compact octal map binary stream, encoding free space and occupied space. The complete message contains the complete probabilities and all additional data stored in the tree.
occupied_cells_vis_array	visualization_msgs/MarkerArray	In RViz, all occupied voxels are marked as visualization "boxes"

Subscription topic	Type	Description
octomap_point_cloud_centers	sensor_msgs/PointCloud2	The centers of all occupied voxels serve as point clouds, useful for visualization. Note that there will be gaps as points have no volume size and octagonal voxels can have different resolutions!
map	nav_msgs/OccupancyGrid	Project a 2D occupancy map downward from a 3D map.
Service	**Type**	**Description**
octomap_binary	octomap_msgs/GetOctomap	The complete maximum likelihood occupancy map as a compact binary stream of octal maps, encoding free space and occupied space.
clear_bbx	octomap_msgs/BoundingBoxQuery	Clears an area in the 3D occupancy map, setting all voxels in the area to "free"
reset	std_srvs/Empty	Reset the entire map

Node view

```
rqt_graph
```



9.3.2. Configuration parameters

Parameters	Type	Default value	Description
frame_id	string	/map	The static global frame in which the map will be published. When building a map dynamically, sensor data needs to be converted to this frame.
resolution	float	0.05	The resolution of the map in meters when starting from an empty map. Otherwise the resolution of the loaded file will be used.
base_frame_id	string	base_footprint	The robot base that performs ground plane detection (if enabled)
height_map	bool	true	Whether the visualization should encode the height with different colors
color/[r/g/b/a]	float		When ~heigh_map=False, display the color of occupied cells in the [0:1] range
sensor_model/max_range	float	-1 (unlimited)	The maximum range (in meters) for inserting point cloud data when building a map dynamically. Limiting the range to a useful range (e.g. 5 meters) prevents false error points away from the robot.
sensor_model/[hit|miss]	float	0.7 / 0.4	Hit and miss probabilities in the sensor model when building the map dynamically

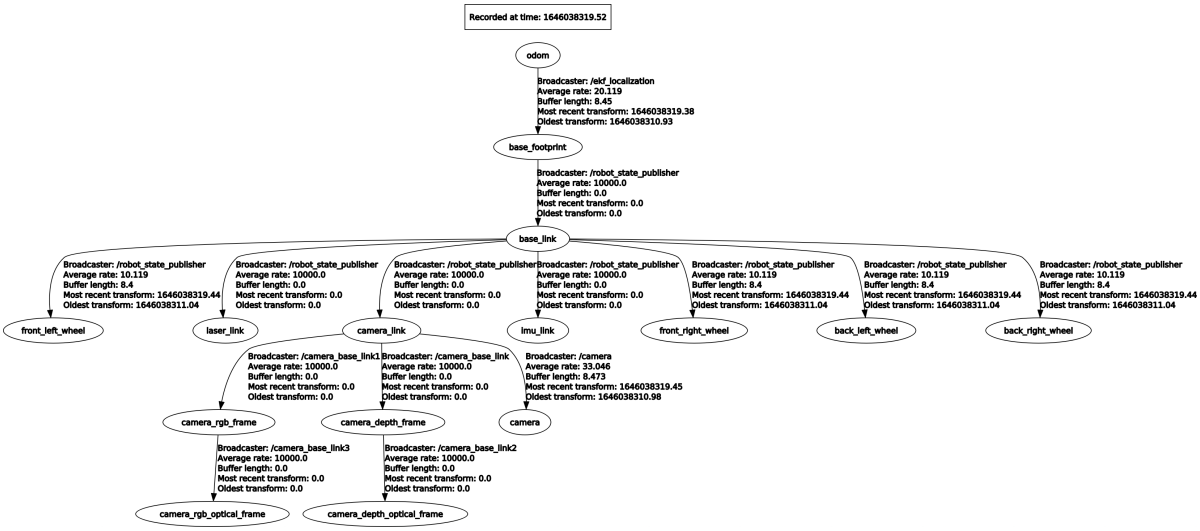
Parameters	Type	Default value	Description
sensor_model/[min|max]	float	0.12 / 0.97	Minimum and maximum probability of clamping when building a map dynamically
latch	bool	True for static mapping, false if no initial mapping is given	Whether the topic is locked for publishing or only published once per change. For best performance when building maps (which update frequently), set this to false. When set to true, all topics and visualizations will be created on all changes on each map.
filter_ground	bool	false	Whether the ground plane should be detected and ignored from scan data when dynamically building a map using pcl::SACMODEL_vertical_plane. This clears everything on the ground, but does not insert the ground into the map as an obstacle. If this feature is enabled, the ~ground_filter/... parameters can be further configured.
ground_filter/distance	float	0.04	The distance threshold of the point (z direction) to be segmented to the ground plane
ground_filter/angle	float	0.15	The angle threshold between the detected plane and the horizontal plane detected as the ground
ground_filter/plane_distance	float	0.07	Distance threshold at z=0 for a plane to be detected as ground (fourth coefficient of the PCL plane equation)
pointcloud*[min|max]*z	float	-/+ infinity	Insert the minimum and maximum height to be considered in the callback.
occupancy*[min|max]*z	float	-/+ infinity	The minimum and maximum height to be considered in the final map.

9.3.3, TF transformation

Required TF transformation	Description
data sensor frame --> /map	If scanning integration is done, the sensor data needs to be converted into a global map frame. This information needs to be obtained from external SLAM or localization nodes.

View tf tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



9.4. Expansion testing

Note: This case is a test version and is related to device performance. If the performance is too low, it may not start properly.

Start the underlying driver + orb_slam (virtual machine side)

```
roslaunch yahboomcar_slam test_orb_slam.launch
```

- [bUseViewer] parameter: whether to open the visualization window of orbslam.

Start octree mapping (virtual machine side)

```
roslaunch yahboomcar_slam test_pcl_mapping.launch
```

- [use_viewer] parameter: whether to open the visualization window of pcl_viewer.
- [local_frame_id] parameter: local map coordinate system.
- [global_frame_id] parameter: global map coordinate system.

After opening, the [viewer] window will pop up, as shown below

Move the camera slowly, when a picture appears, as shown below

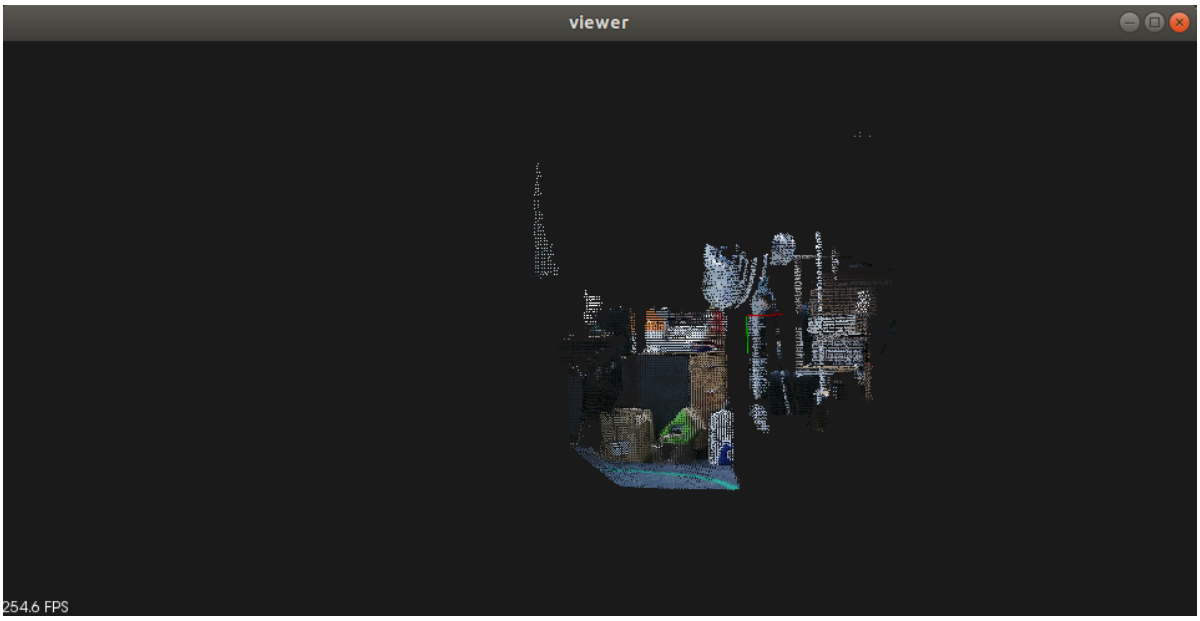
The coordinate system needs to be scaled and rotated as shown in the figure below

Sliding wheel: zoom in or out

Press and hold scroll wheel: pan

Left mouse button: rotate

Right mouse button: zoom in or out



Slowly move the camera to build the map as shown below. After the build is completed, [Ctrl+c] closes and the pcd point cloud file is automatically saved. The path file name under this function package is [resultPointCloudFile.pcd]

pcl_viewer installation command

```
sudo apt-get install pcl-tools
```

View and enter the directory where the pcd file is located

```
pcl_viewer resultPointCloudFile.pcd
```