7. AR QR code

7.1. Overview

wiki: http://wiki.ros.org/ar_track_alvar/

Source code: https://github.com/ros-perception/ar_track_alvar.git

Function package location: ~/transbot_ws/src/transbot_visual

ARTag (AR tag, AR means "augmented reality") is a benchmark marking system, which can be understood as a reference for other objects. It looks similar to a QR code, but its encoding system and QR code are still very different. The difference is mostly used in camera calibration, robot positioning, augmented reality (AR) and other applications. One of the most important functions is to identify the pose relationship between the object and the camera. ARTag can be attached to the object, or ARTag label can be attached to the plane to calibrate the camera. After the camera recognizes the ARTag, it can calculate the position and attitude of the tag in the camera coordinates.

ar_track_alvar has 4 main functions:

- Generate AR tags with different sizes, resolutions and data/ID encodings.
- Recognize and track poses of individual AR tags, optionally integrating kinect depth data (when kinect is available) for better pose estimation.
- Recognize and track poses in "bundles" consisting of multiple tags. This allows for more stable pose estimation, robustness to occlusions, and tracking of multilateral objects.
- Automatically calculate spatial relationships between tags in bundles using camera images so users don't have to manually measure and enter tag locations in an XML file to use the bundle feature.

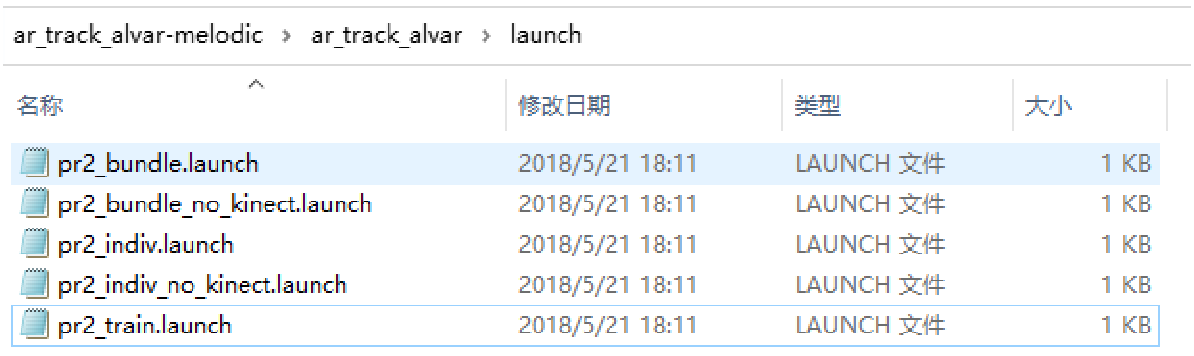
Alvar is newer and more advanced than ARToolkit, which has been the basis for several other ROS AR tag packages. Alvar features adaptive thresholding to handle various lighting conditions, optical flow-based tracking for more stable pose estimation, and an improved tag recognition method that does not slow down significantly as the number of tags increases.

7.2. Create ARTag

7.2.1. Install software package

The factory image of the car has already been installed, no need to reinstall it.

```
sudo apt install ros-melodic-ar-track-alvar
```



ar_track_alvar-melodic › ar_track_alvar › launch

名称	修改日期	类型	大小
pr2_bundle.launch	2018/5/21 18:11	LAUNCH 文件	1 KB
pr2_bundle_no_kinect.launch	2018/5/21 18:11	LAUNCH 文件	1 KB
pr2_indiv.launch	2018/5/21 18:11	LAUNCH 文件	1 KB
pr2_indiv_no_kinect.launch	2018/5/21 18:11	LAUNCH 文件	1 KB
pr2_train.launch	2018/5/21 18:11	LAUNCH 文件	1 KB

ar_track_alvar is an open source marker library that provides pr2+kinect examples. The first use case of this package is to identify and track gestures from (possibly) multiple AR tags, each considered individually.

7.2.2. Create AR QR code

- Continuously generate multiple tags on one image

  jetson/Raspberry Pi 4B

```
rosrun ar_track_alvar createMarker
```

Raspberry Pi 5

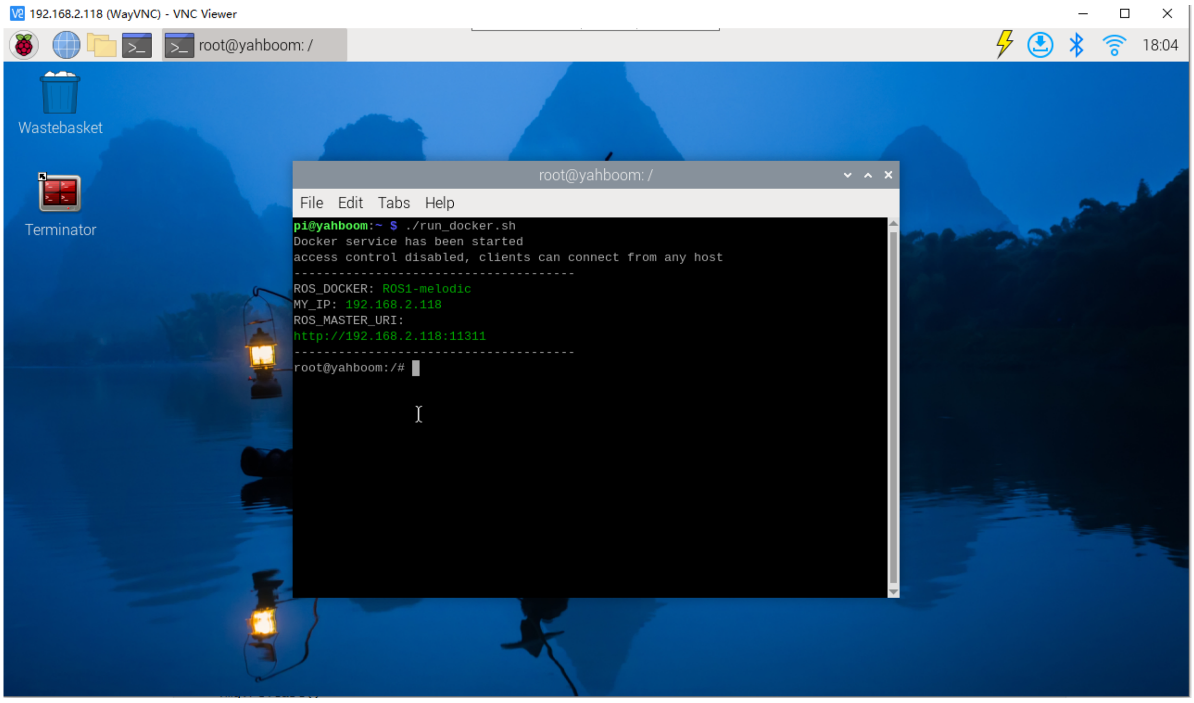
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If there is a terminal that automatically starts docker, or there is a docker terminal that has been opened, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

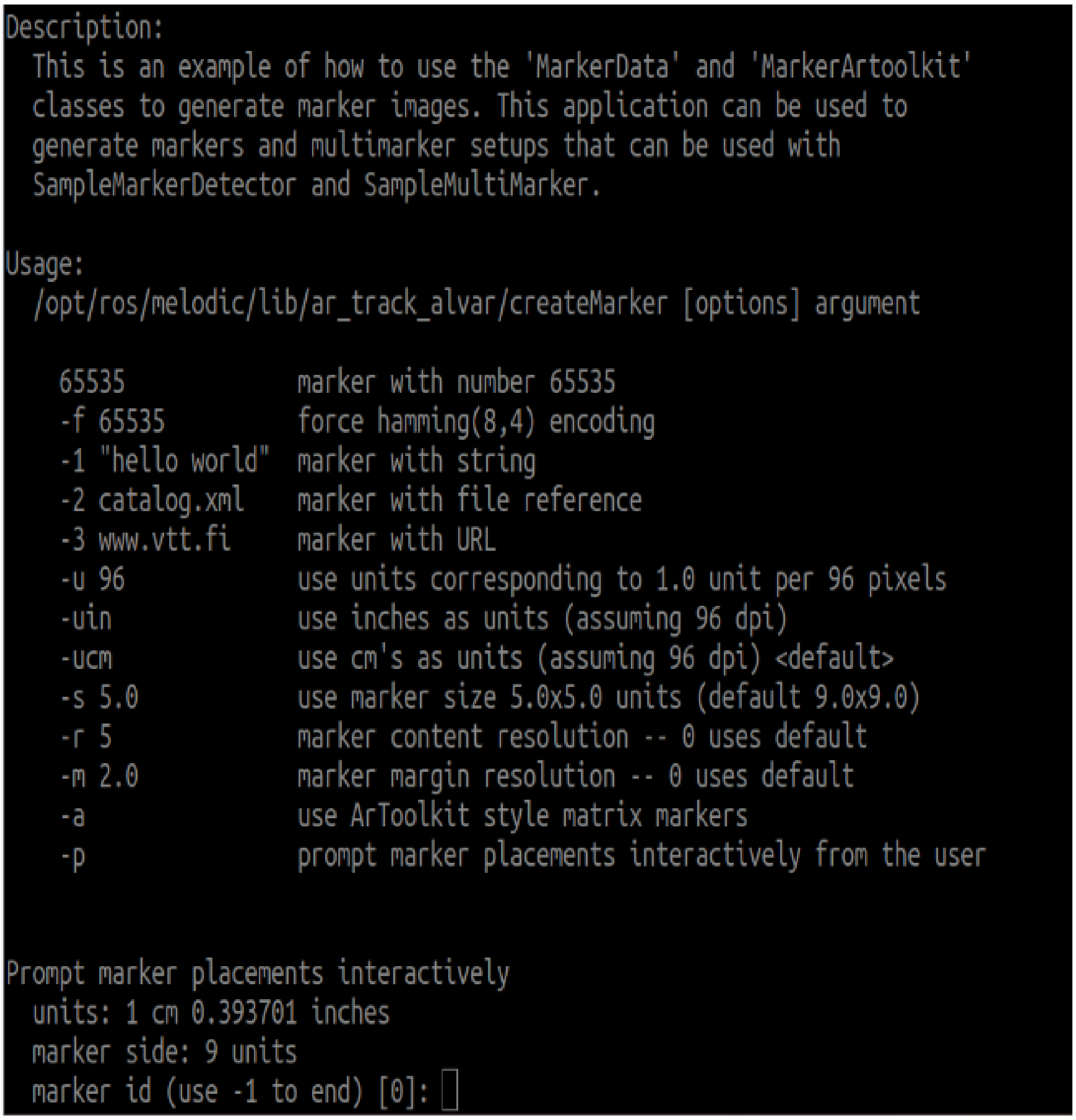
Start docker manually

```
./run_docker.sh
```



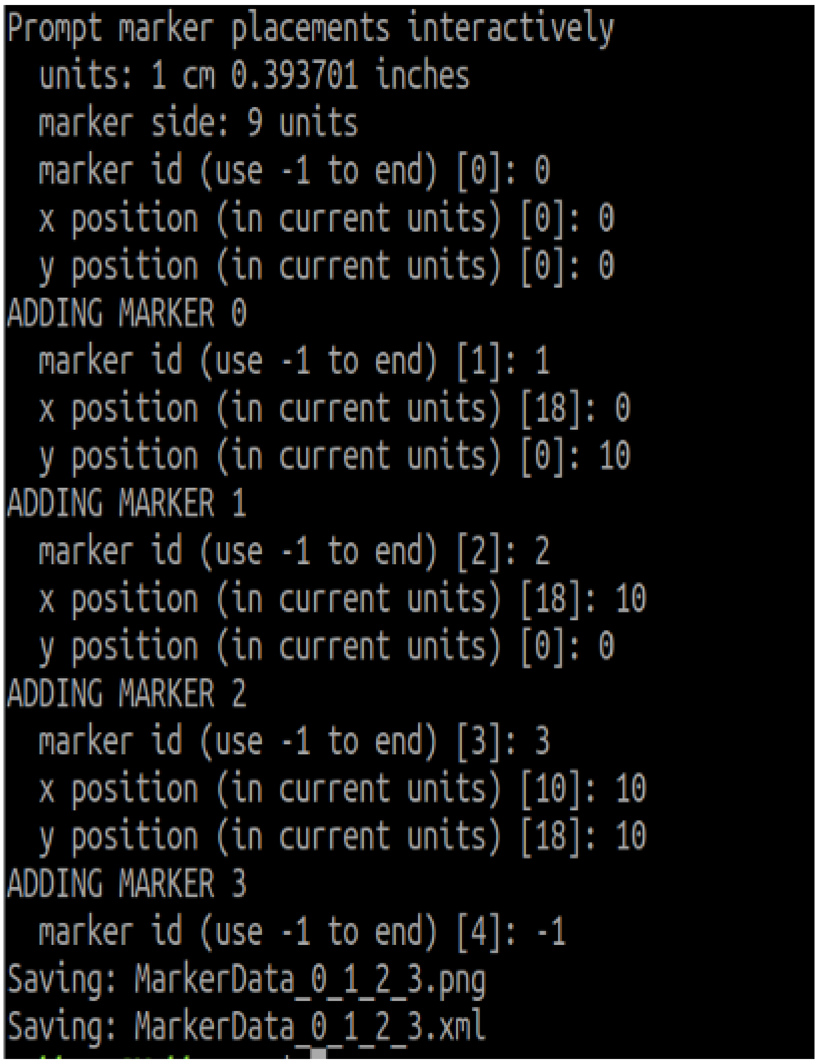
```
pi@yahboom:~ $ ./run_docker.sh
Docker service has been started
access control disabled, clients can connect from any host
--------------------------------------
ROS_DOCKER: ROS1-melodic
MY_IP: 192.168.2.118
ROS_MASTER_URI:
http://192.168.2.118:11311
--------------------------------------
root@yahboom:/#
```

```
rosrun ar_track_alvar createMarker
```



```
Description:
  This is an example of how to use the 'MarkerData' and 'MarkerArtoolkit'
  classes to generate marker images. This application can be used to
  generate markers and multimarker setups that can be used with
  SampleMarkerDetector and SampleMultiMarker.

Usage:
  /opt/ros/melodic/lib/ar_track_alvar/createMarker [options] argument

    65535             marker with number 65535
    -f 65535          force hamming(8,4) encoding
    -1 "hello world"  marker with string
    -2 catalog.xml    marker with file reference
    -3 www.vtt.fi     marker with URL
    -u 96             use units corresponding to 1.0 unit per 96 pixels
    -uin              use inches as units (assuming 96 dpi)
    -ucm              use cm's as units (assuming 96 dpi) <default>
    -s 5.0            use marker size 5.0x5.0 units (default 9.0x9.0)
    -r 5              marker content resolution -- 0 uses default
    -m 2.0            marker margin resolution -- 0 uses default
    -a                use ArToolkit style matrix markers
    -p                prompt marker placements interactively from the user


Prompt marker placements interactively
  units: 1 cm 0.393701 inches
  marker side: 9 units
  marker id (use -1 to end) [0]:
```

You can enter [ID] and location information here, and enter [-1] to end. One or more can be generated, and the layout can be designed by yourself.



```
Prompt marker placements interactively
  units: 1 cm 0.393701 inches
  marker side: 9 units
  marker id (use -1 to end) [0]: 0
  x position (in current units) [0]: 0
  y position (in current units) [0]: 0
ADDING MARKER 0
  marker id (use -1 to end) [1]: 1
  x position (in current units) [18]: 0
  y position (in current units) [0]: 10
ADDING MARKER 1
  marker id (use -1 to end) [2]: 2
  x position (in current units) [18]: 10
  y position (in current units) [0]: 0
ADDING MARKER 2
  marker id (use -1 to end) [3]: 3
  x position (in current units) [10]: 10
  y position (in current units) [18]: 10
ADDING MARKER 3
  marker id (use -1 to end) [4]: -1
Saving: MarkerData_0_1_2_3.png
Saving: MarkerData_0_1_2_3.xml
```

- Generate a single number

Command + parameters directly generate digital images; for example

jetson/Raspberry Pi 4B

```
rosrun ar_track_alvar createMarker 11
rosrun ar_track_alvar createMarker -s 5 33
```

Raspberry Pi 5

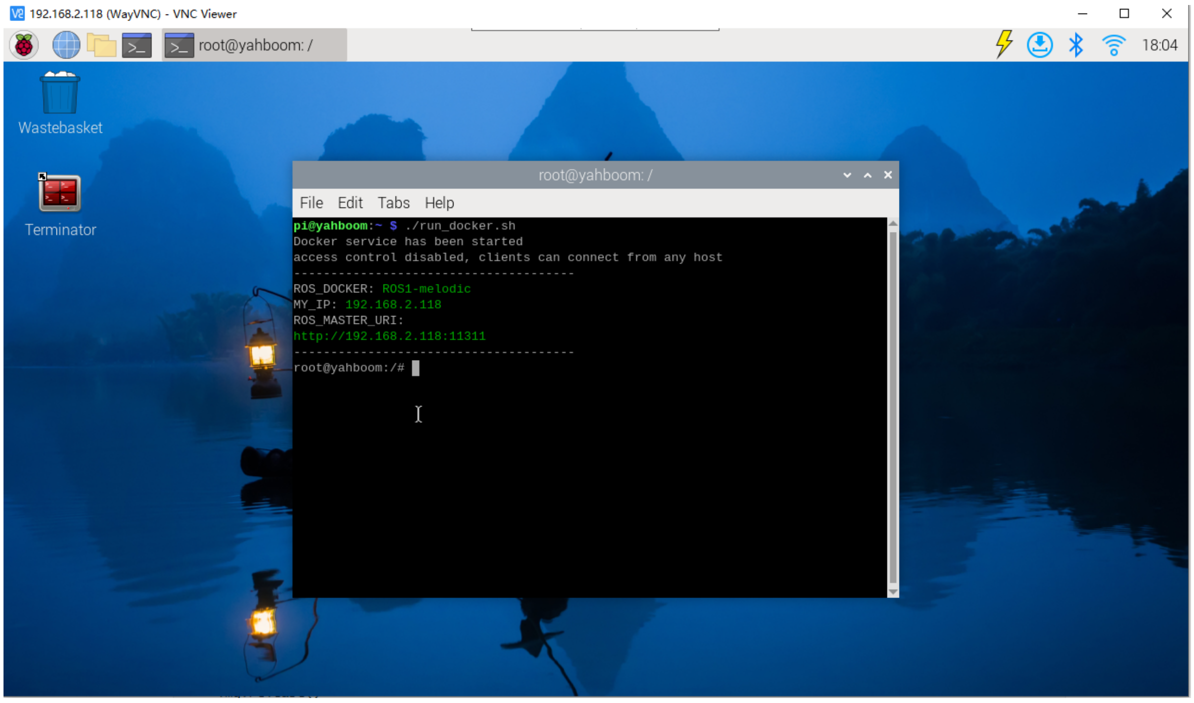
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If there is a terminal that automatically starts docker, or there is a docker terminal that has been opened, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```



```
pi@yahboom:~ $ ./run_docker.sh
Docker service has been started
access control disabled, clients can connect from any host
-------------------------------------
ROS_DOCKER: ROS1-melodic
MY_IP: 192.168.2.118
ROS_MASTER_URI:
http://192.168.2.118:11311
-------------------------------------
root@yahboom:/# 
```

```
rosrun ar_track_alvar createMarker 11
rosrun ar_track_alvar createMarker -s 5 33
```

11: The number is the QR code of 11. -s: Specify image size. 5: 5x5 picture. 33: The number is the QR code of 33.

7.3. ARTag identification

Note: When starting the camera, you need to load the camera calibration file, otherwise it will not be recognized.

7.3.1, ar_track_alvar node

- **Subscribed topic**

Topic name	**Data type**
/camera_info	(sensor_msgs/CameraInfo)
/image_raw	(sensor_msgs/Image)

- **Published Topics**

Topic name	**Data type**
/visualization_marker	(visualization_msgs/Marker)
/ar_pose_marker	(ar_track_alvar/AlvarMarkers)

- **Provided tf Transforms**

Single QR code: camera coordinate system → AR tag coordinate system

Multiple QR codes: Provides transformation from the camera coordinate system to each AR tag coordinate system (named ar_marker_x), where x is the ID number of the marker.

7.3.2. Launch file analysis

```
<launch>
     <arg name="camDevice" default="USBCam" doc="camDevice type [USBCam]"/>
     <arg name="open_rviz" default="true"/>
     <arg name="marker_size" default="5.0"/>
     <arg name="max_new_marker_error" default="0.08"/>
     <arg name="max_track_error" default="0.2"/>
     <arg name="cam_image_topic" default="/usb_cam/image_raw" if="$(eval
arg('camDevice') == 'USBCam')"/>
     <arg name="cam_info_topic" default="/usb_cam/camera_info" if="$(eval
arg('camDevice') == 'USBCam')"/>
     <arg name="output_frame" default="/usb_cam" if="$(eval arg('camDevice') ==
'USBCam')"/>
     <include if="$(eval arg('camDevice') == 'USBCam')" file="$(find
usb_cam)/launch/usb_cam-test.launch"/>
     <node if="$(eval arg('camDevice') == 'USBCam')" pkg="tf"
type="static_transform_publisher" name="world_to_cam"
           args="0 0 0.5 0 1.57 0 world usb_cam 10"/>
     <node name="ar_track_alvar" pkg="ar_track_alvar"
type="individualMarkersNoKinect" respawn="false" output="screen">
         <param name="marker_size" type="double" value="$(arg marker_size)"/>
         <param name="max_new_marker_error" type="double" value="$(arg
max_new_marker_error)"/>
         <param name="max_track_error" type="double" value="$(arg
max_track_error)"/>
         <param name="output_frame" type="string" value="$(arg output_frame)"/>
         <remap from="camera_image" to="$(arg cam_image_topic)"/>
         <remap from="camera_info" to="$(arg cam_info_topic)"/>
     </node>
     <group if="$(arg open_rviz)">
         <node pkg="rviz" type="rviz" name="rviz" args="-d $(find
transbot_visual)/rviz/ar_track.rviz"/>
     </group>
</launch>
```

Node parameters:

- marker_size (double): Width (in centimeters) of one side of the black square marker border.
- max_new_marker_error (double): Threshold for determining when a new marker can be detected under uncertain conditions.
- max_track_error (double): A threshold that determines how many tracking errors can be observed before the marker disappears.
- camera_image (string): Provides the image topic name used to detect AR tags. This can be monochrome or color, but should be an uncorrected image since correction is done in this package.

- camera_info (string): Subject name that provides camera calibration parameters for correcting images.
- output_frame (string): Publish the coordinate position of the AR tag in the camera coordinate system.

7.3.3. Start identification instance

- **jetson/Raspberry Pi 4B**

  ```
  roslaunch transbot_visual ar_track.launch camDevice:=USBCam
  ```

  Raspberry Pi 5

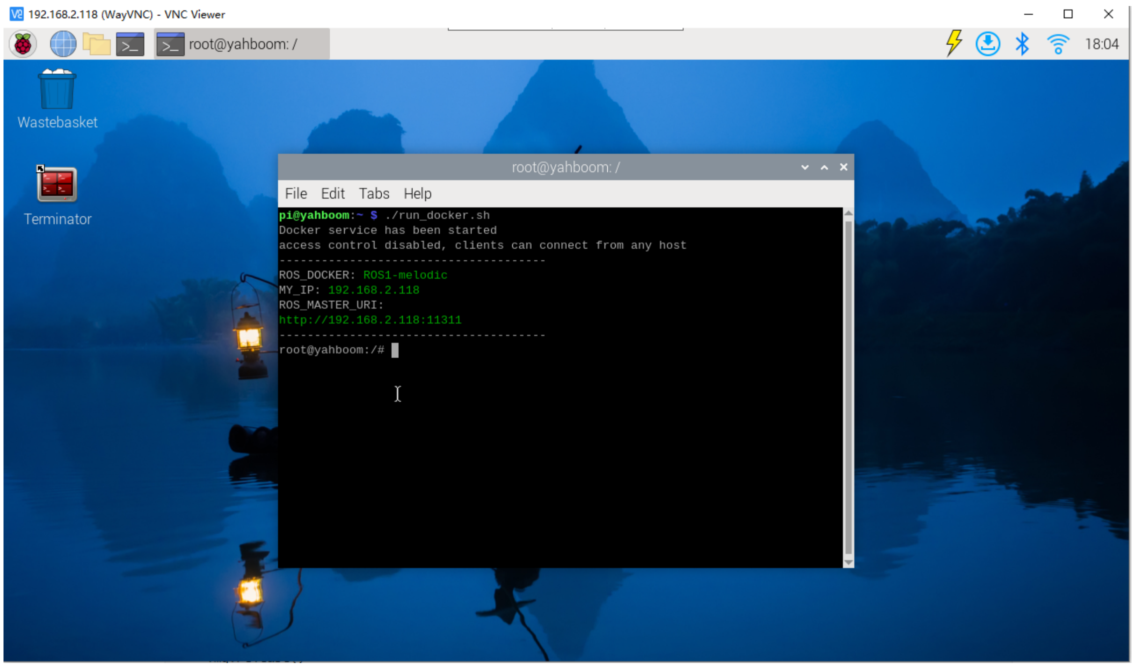
  Before running, please confirm that the large program has been permanently closed

  Enter docker

  Note: If there is a terminal that automatically starts docker, or there is a docker terminal that has been opened, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

  Start docker manually

  ```
  ./run_docker.sh
  ```

  ```
  pi@yahboom:~ $ ./run_docker.sh
  Docker service has been started
  access control disabled, clients can connect from any host
  --------------------------------------
  ROS_DOCKER: ROS1-melodic
  MY_IP: 192.168.2.118
  ROS_MASTER_URI:
  http://192.168.2.118:11311
  --------------------------------------
  root@yahboom:/#
  ```

  ```
  roslaunch transbot_visual ar_track.launch camDevice:=USBCam
  ```

  - camDevice: USBCam

  In rviz, you need to set the corresponding camera topic name.

  - Image_Topic: The high frame rate camera is [/usb_cam/image_raw].
  - Marker: The display component of rviz. Different squares display the location of the AR QR code.
  - TF: The display component of rviz, used to display the coordinate system of AR QR codes.
  - Camera: The display component of rviz, which displays the camera screen.

- world: world coordinate system.
- usb_cam/camera_link: camera coordinate system.

7.3.4. View node graph

jetson/Raspberry Pi 4B/Raspberry Pi 4B

```
rqt_graph
```

Raspberry Pi 5

Enter the same docker from multiple terminals

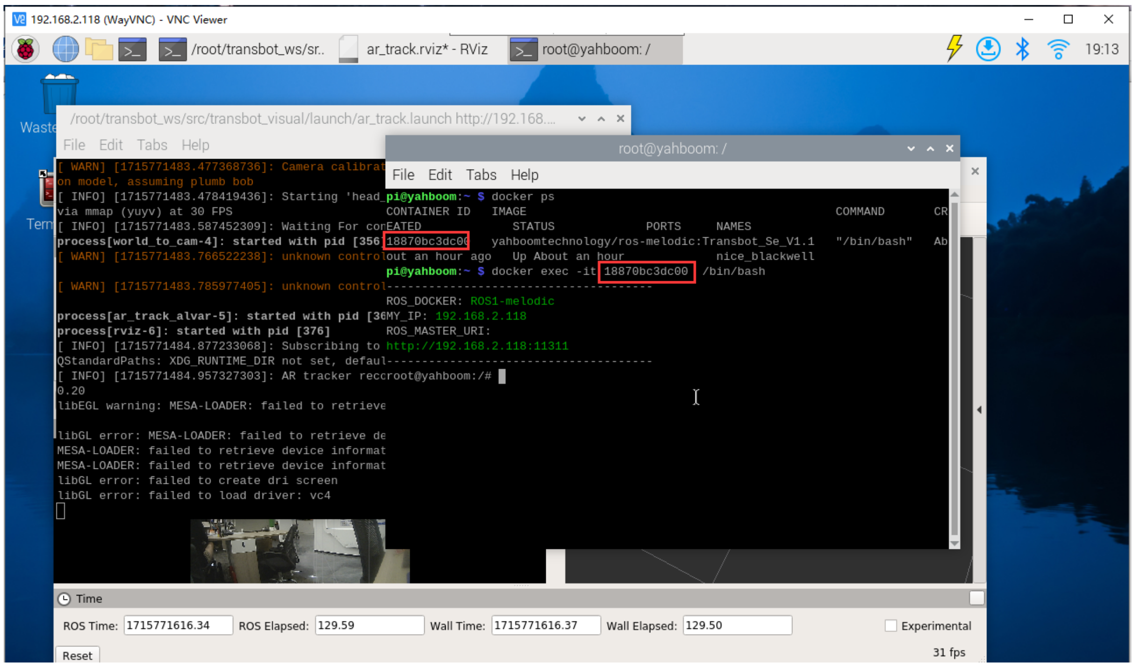
Keep the program of the previous docker terminal running and open a new terminal

Enter the following command

```
docker ps
```

Enter the same docker and use the following 18870bc3dc00 to modify the ID displayed on the actual terminal.

```
docker exec -it 18870bc3dc00 /bin/bash
```

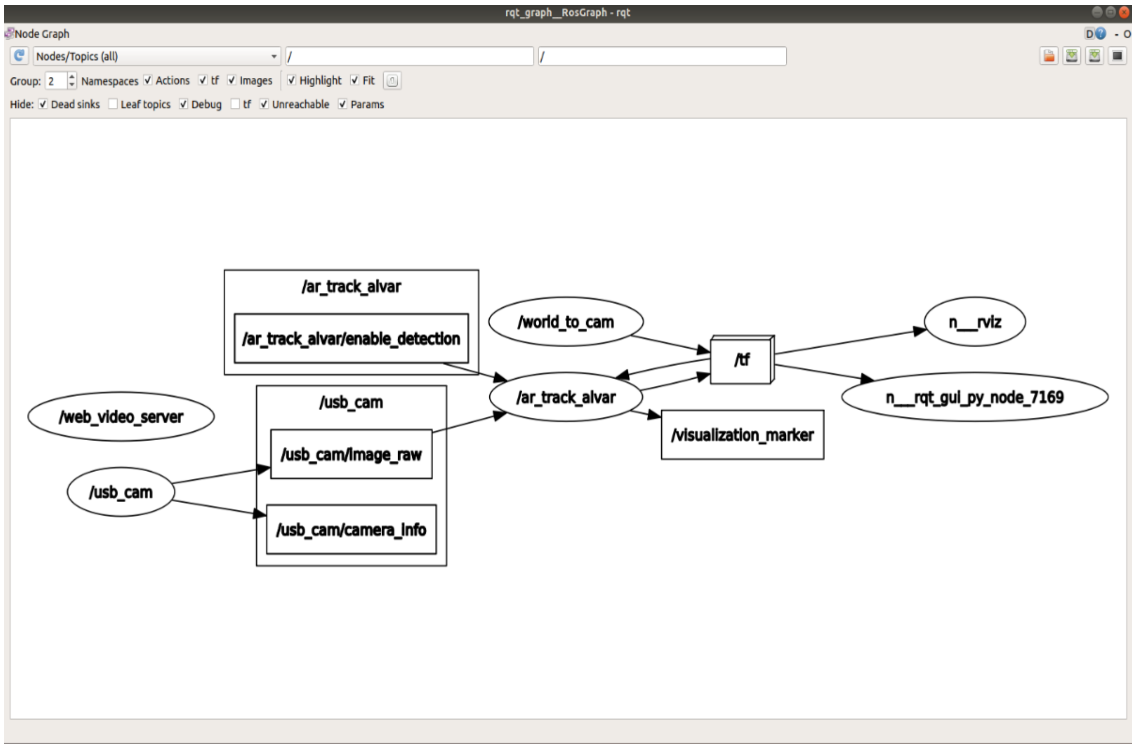


```
[ WARN] [1715771483.477368736]: Camera calibrat
on model, assuming plumb bob
[ INFO] [1715771483.478419436]: Starting 'head_
via mmap (yuyv) at 30 FPS
[ INFO] [1715771483.587452309]: Waiting For con
process[world_to_cam-4]: started with pid [356]
[ WARN] [1715771483.766522238]: unknown control

[ WARN] [1715771483.785977405]: unknown control

process[ar_track_alvar-5]: started with pid [36
process[rviz-6]: started with pid [376]
[ INFO] [1715771484.877233068]: Subscribing to
QStandardPaths: XDG_RUNTIME_DIR not set, defaul
[ INFO] [1715771484.957327303]: AR tracker recc
0.20
libEGL warning: MESA-LOADER: failed to retrieve

libGL error: MESA-LOADER: failed to retrieve de
MESA-LOADER: failed to retrieve device informat
MESA-LOADER: failed to retrieve device informat
libGL error: failed to create dri screen
libGL error: failed to load driver: vc4
```

```
pi@yahboom:~ $ docker ps
CONTAINER ID   IMAGE                                              COMMAND       CR
EATED          STATUS              PORTS     NAMES
18870bc3dc00   yahboomtechnology/ros-melodic:Transbot_Se_V1.1   "/bin/bash"   Ab
out an hour ago   Up About an hour             nice_blackwell
pi@yahboom:~ $ docker exec -it 18870bc3dc00  /bin/bash
--------------------------------------------
ROS_DOCKER: ROS1-melodic
MY_IP: 192.168.2.118
ROS_MASTER_URI:
http://192.168.2.118:11311
--------------------------------------------
root@yahboom:/#
```

ROS Time: 1715771616.34 ROS Elapsed: 129.59 Wall Time: 1715771616.37 Wall Elapsed: 129.50

```
rqt_graph
```



7.3.5. View tf tree

jetson/Raspberry Pi 4B/Raspberry Pi 4B

```
rosrun rqt_tf_tree rqt_tf_tree
```

Raspberry Pi 5

Enter the same docker from multiple terminals

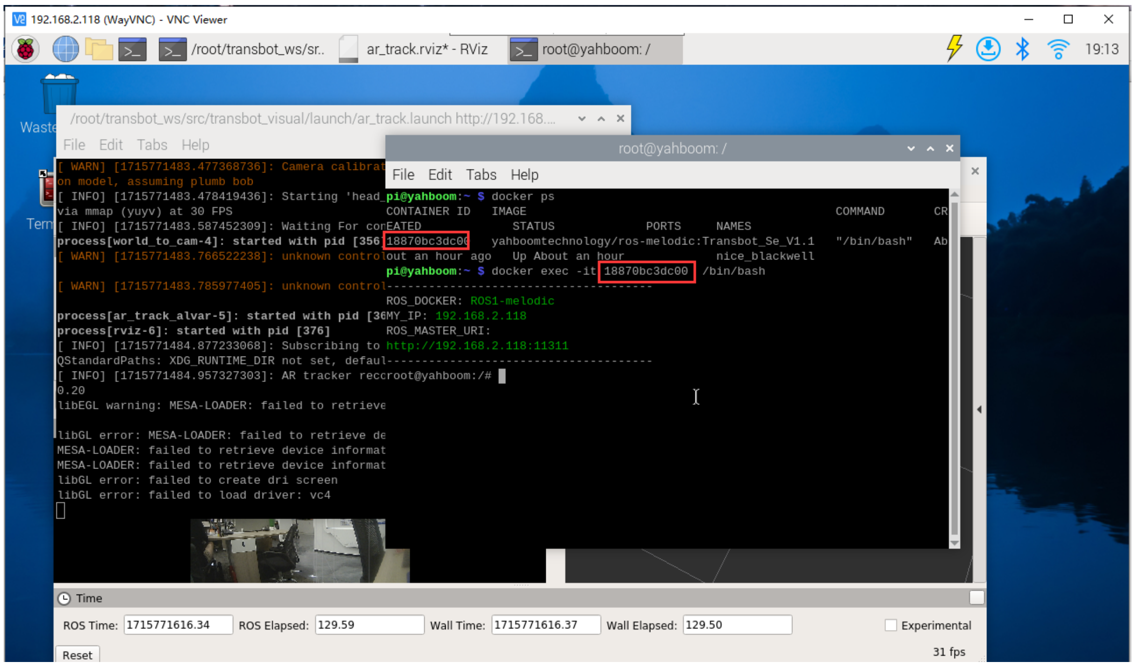
Keep the program of the previous docker terminal running and open a new terminal

Enter the following command

```
docker ps
```

Enter the same docker and use the following 18870bc3dc00 to modify the ID displayed on the actual terminal.

```
docker exec -it 18870bc3dc00 /bin/bash
```


```
[ WARN] [1715771483.477368736]: Camera calibrat
on model, assuming plumb bob
[ INFO] [1715771483.478419436]: Starting 'head_
via mmap (yuyv) at 30 FPS
[ INFO] [1715771483.587452309]: Waiting For con
process[world_to_cam-4]: started with pid [356]
[ WARN] [1715771483.766522238]: unknown control

[ WARN] [1715771483.785977405]: unknown control

process[ar_track_alvar-5]: started with pid [36
process[rviz-6]: started with pid [376]
[ INFO] [1715771484.877233068]: Subscribing to
QStandardPaths: XDG_RUNTIME_DIR not set, defaul
[ INFO] [1715771484.957327303]: AR tracker recc
0.20
libEGL warning: MESA-LOADER: failed to retrieve

libGL error: MESA-LOADER: failed to retrieve de
MESA-LOADER: failed to retrieve device informat
MESA-LOADER: failed to retrieve device informat
libGL error: failed to create dri screen
libGL error: failed to load driver: vc4
```

```
pi@yahboom:~ $ docker ps
CONTAINER ID   IMAGE                                          COMMAND       CR
EATED             STATUS              PORTS     NAMES
18870bc3dc00   yahboomtechnology/ros-melodic:Transbot_Se_V1.1   "/bin/bash"   Ab
out an hour ago   Up About an hour              nice_blackwell
pi@yahboom:~ $ docker exec -it 18870bc3dc00 /bin/bash
--------------------------------------------
ROS_DOCKER: ROS1-melodic
MY_IP: 192.168.2.118
ROS_MASTER_URI:
http://192.168.2.118:11311
--------------------------------------------
root@yahboom:/#
```

ROS Time: 1715771616.34 ROS Elapsed: 129.59 Wall Time: 1715771616.37 Wall Elapsed: 129.50 Experimental

192.168.2.118 (WayVNC) - VNC Viewer
/root/transbot_ws/sr..
ar_track.rviz* - RViz
root@yahboom: /
19:13
/root/transbot_ws/src/transbot_visual/launch/ar_track.launch http://192.168....
File Edit Tabs Help
Time
Reset
31 fps

```
rosrun rqt_tf_tree rqt_tf_tree
```

Through rviz, we can intuitively see the relative position of the QR code and the camera. The camera and world coordinate system are set by yourself.

7.3.6. View output information

jetson/Raspberry Pi 4B/Raspberry Pi 4B

```
rostopic echo /ar_pose_marker
```

Raspberry Pi 5

Enter the same docker from multiple terminals

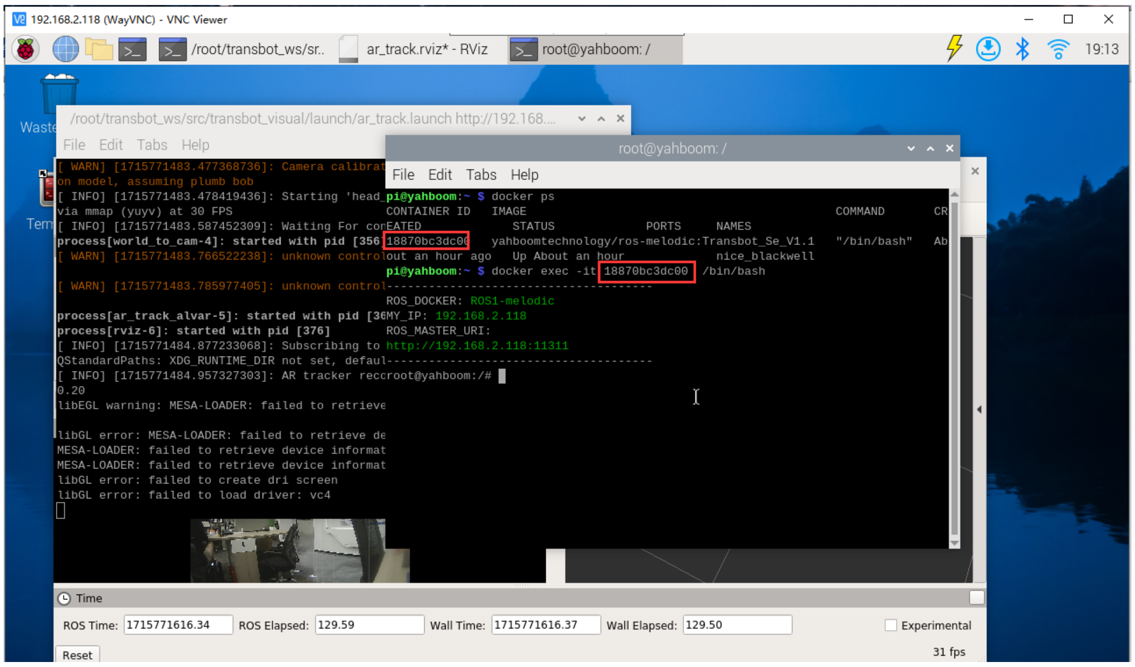
Keep the program of the previous docker terminal running and open a new terminal

Enter the following command

```
docker ps
```

Enter the same docker and use the following 18870bc3dc00 to modify the ID displayed on the actual terminal.

```
docker exec -it 18870bc3dc00 /bin/bash
```



```
[ WARN] [1715771483.477368736]: Camera calibrat
on model, assuming plumb bob
[ INFO] [1715771483.478419436]: Starting 'head_
via mmap (yuyv) at 30 FPS
[ INFO] [1715771483.587452309]: Waiting For con
process[world_to_cam-4]: started with pid [356]
[ WARN] [1715771483.766522238]: unknown control

[ WARN] [1715771483.785977405]: unknown control

process[ar_track_alvar-5]: started with pid [36
process[rviz-6]: started with pid [376]
[ INFO] [1715771484.877233068]: Subscribing to
QStandardPaths: XDG_RUNTIME_DIR not set, defaul
[ INFO] [1715771484.957327303]: AR tracker recc
0.20
libEGL warning: MESA-LOADER: failed to retrieve

libGL error: MESA-LOADER: failed to retrieve de
MESA-LOADER: failed to retrieve device informat
MESA-LOADER: failed to retrieve device informat
libGL error: failed to create dri screen
libGL error: failed to load driver: vc4
```

```
pi@yahboom:~ $ docker ps
CONTAINER ID   IMAGE                                         COMMAND       CR
EATED              STATUS              PORTS     NAMES
18870bc3dc00   yahboomtechnology/ros-melodic:Transbot_Se_V1.1   "/bin/bash"   Ab
out an hour ago   Up About an hour              nice_blackwell
pi@yahboom:~ $ docker exec -it 18870bc3dc00 /bin/bash
--------------------------------------------
ROS_DOCKER: ROS1-melodic
MY_IP: 192.168.2.118
ROS_MASTER_URI:
http://192.168.2.118:11311
--------------------------------------------
root@yahboom:/#
```

ROS Time: 1715771616.34 ROS Elapsed: 129.59 Wall Time: 1715771616.37 Wall Elapsed: 129.50

```
rostopic echo /ar_pose_marker
```

Displayed as follows:

```
    header:
      seq: 0
      stamp:
        secs: 1630584915
        nsecs: 196221070
      frame_id: "/usb_cam"
    ID: 3
    confidence: 0
    pose:
      header:
        seq: 0
        stamp:
          secs: 0
          nsecs: 0
        frame_id: ''
      pose:
        position:
          x: 0.0249847882514
          y: 0.0290736736336
          z: 0.218054183012
        orientation:
          x: 0.682039034537
          y: 0.681265739969
          z: -0.156112715404
          w: 0.215240718735
```

- frame_id: The coordinate system name of the camera
- id: The number recognized is 3
- pose: the pose of the QR code
- position: the position of the QR code coordinate system relative to the camera coordinate system

- orientation: the attitude of the QR code coordinate system relative to the camera coordinate system