5.AR QR code

The Raspberry Pi motherboard does not currently support running this function on the motherboard, but the image data can be transmitted to the virtual machine through distributed communication, and the virtual machine can track and identify it. The premise is that the motherboard and the virtual machine need to set up distributed communication, refer to the previous tutorial to set up.

2.1 Overview

ARTag (AR tag, AR means "augmented reality") is a benchmark marking system that can be understood as a reference for other objects.It looks similar to a QR code, but its encoding system is still very different from a QR code. It is mostly used in camera calibration, robot positioning, augmented reality (AR) and other applications.One of the most important functions is to identify the position 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:

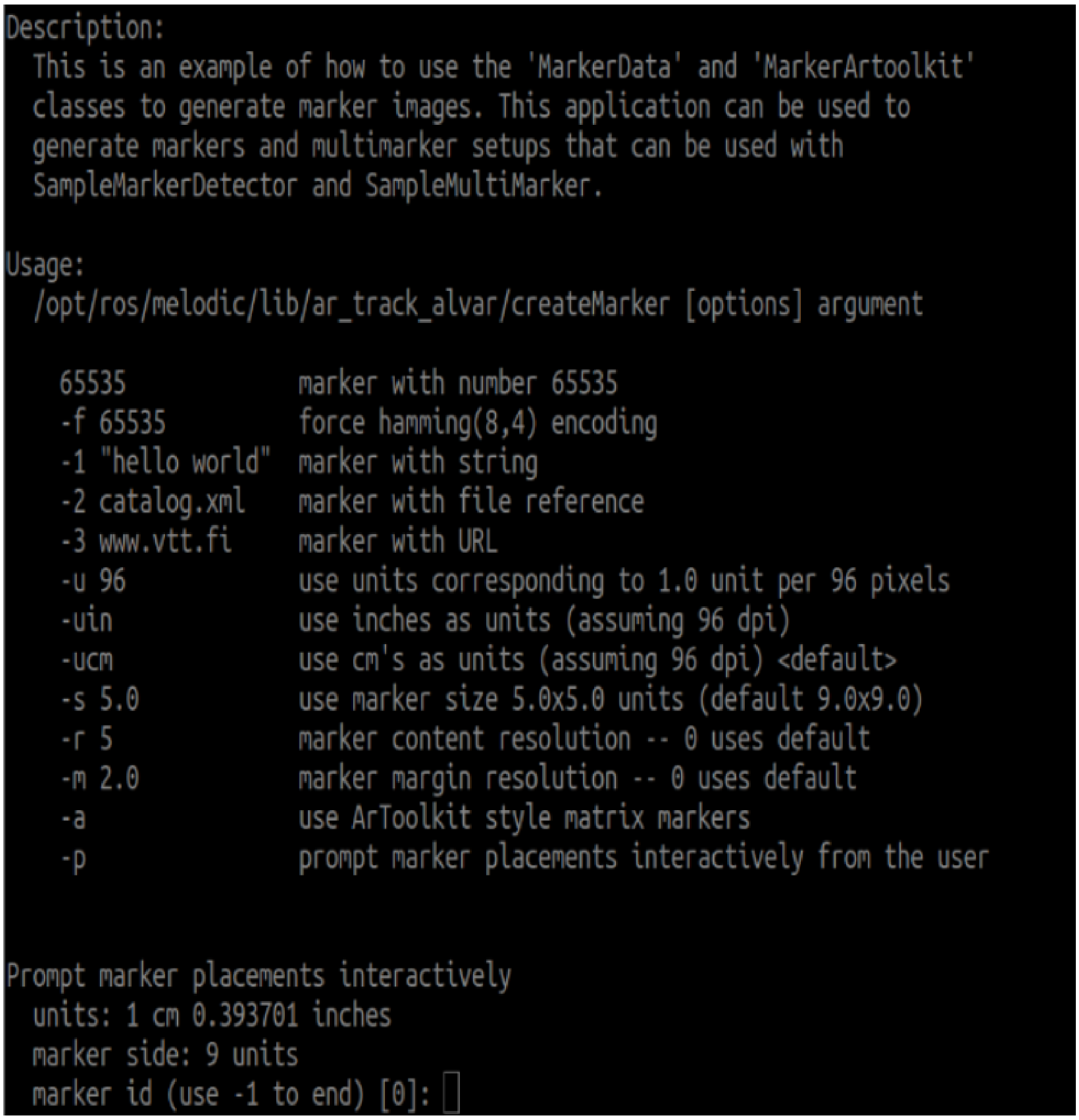
- Generate AR tags with different sizes, resolutions and data/ID encodings.
- Recognize and track poses of individual AR tags, optionally integrating kinect depth data (when kinect is available) for better pose estimation.
- Recognize and track poses in "bundles" consisting of multiple tags. This allows 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.

2.2 Create AR QR code

- Continuously generate multiple tags on one image

```
roscore
rosrun ar_track_alvar createMarker
```



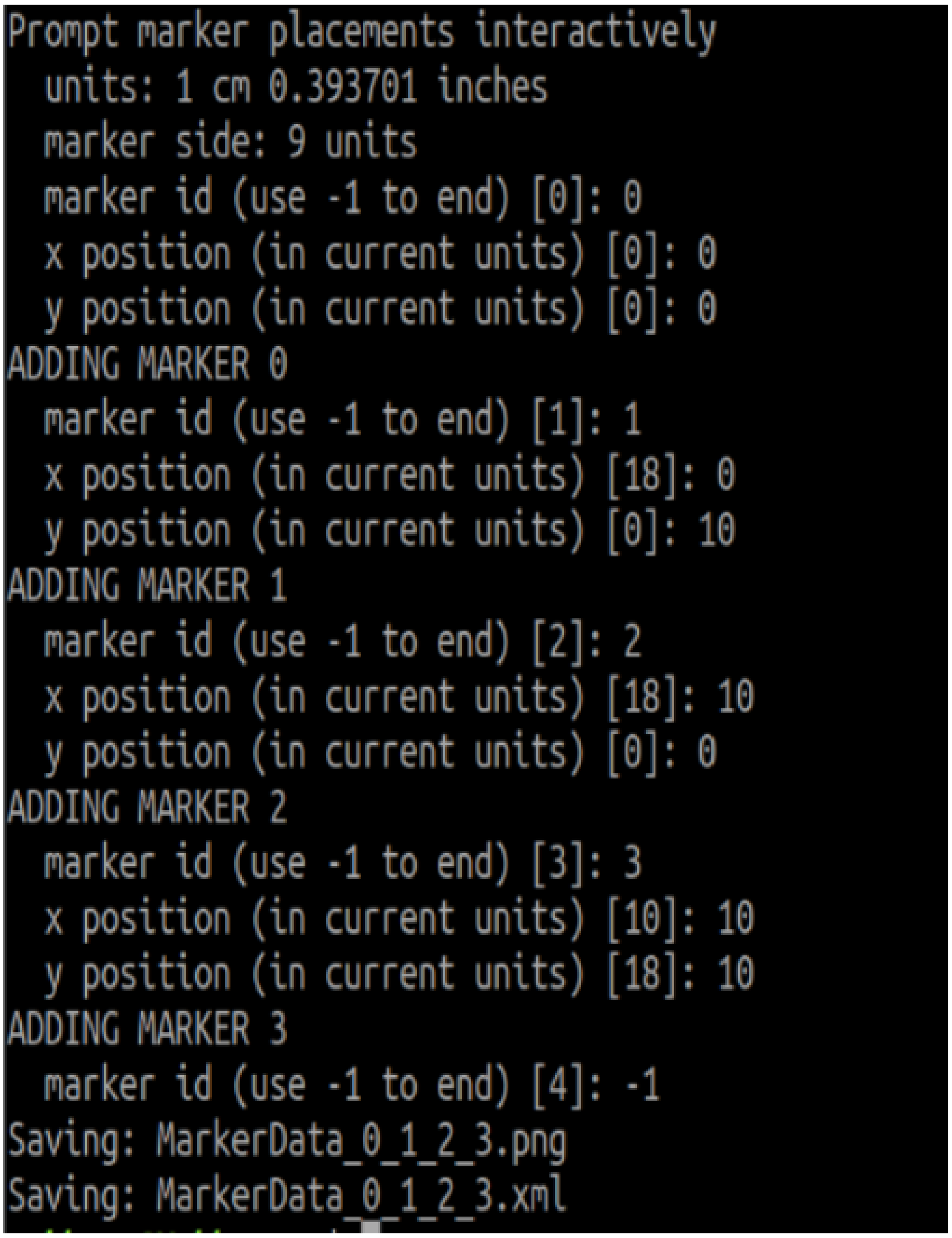
```
Description:
  This is an example of how to use the 'MarkerData' and 'MarkerArtoolkit'
  classes to generate marker images. This application can be used to
  generate markers and multimarker setups that can be used with
  SampleMarkerDetector and SampleMultiMarker.

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 a digital picture; for example

  ```
  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.

The files generated by the above two generation methods are stored in the terminal directory of the running command.

2.3 ARTag identification

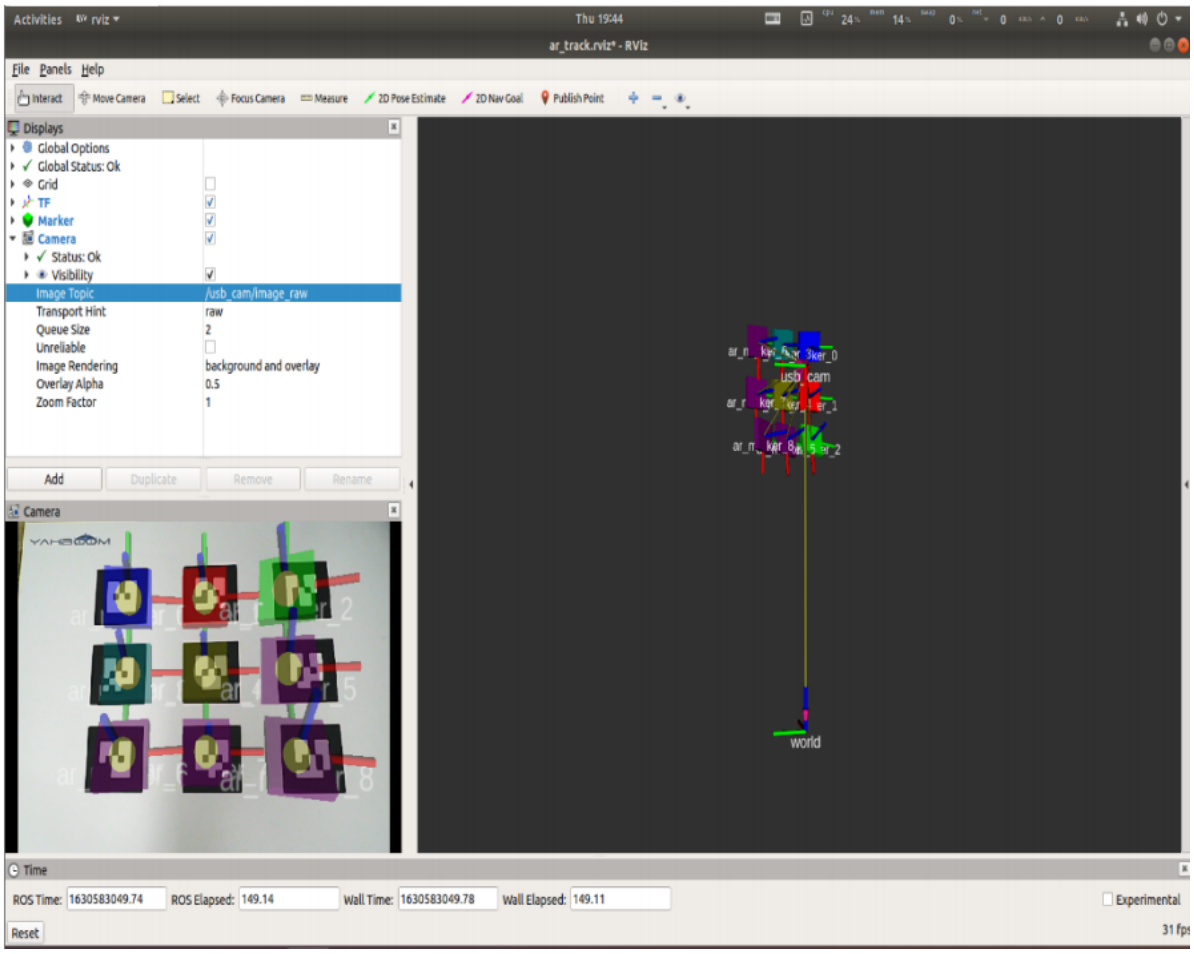
Raspberry Pi motherboard startup method is as follows:

```
#Establishing multi-machine communication between Raspberry Pi and virtual machine, and taking Raspberry Pi as the host as an example
roslaunch usb_cam usb_cam-test.launch   #Start the Raspberry Pi and drive the USB camera
roslaunch dofbot_visual ar_track.launch open_rviz:=true #Start the virtual machine, identify and track
```

Jetson-nano and the virtual machine are started as follows,

```
roslaunch dofbot_visual ar_track.launch open_rviz:=true
```

- The open_rviz parameter is open by default



Displays
Global Options
Global Status: Ok
Grid ☐
TF ✓
Marker ✓
Camera ✓
Status: Ok
Visibility ✓
Image Topic /usb_cam/image_raw
Transport Hint raw
Queue Size 2
Unreliable ☐
Image Rendering background and overlay
Overlay Alpha 0.5
Zoom Factor 1

Time
ROS Time: 1630583049.74 ROS Elapsed: 149.14 Wall Time: 1630583049.78 Wall Elapsed: 149.11 ☐ Experimental

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

- Image_Topic: The camera topic 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 coordinate system.

2.4 ar_track_alvar node

Subscribed topics:

话题名	数据类型
/camera_info	(sensor_msgs/CameraInfo)
/image_raw	(sensor_msgs/Image)

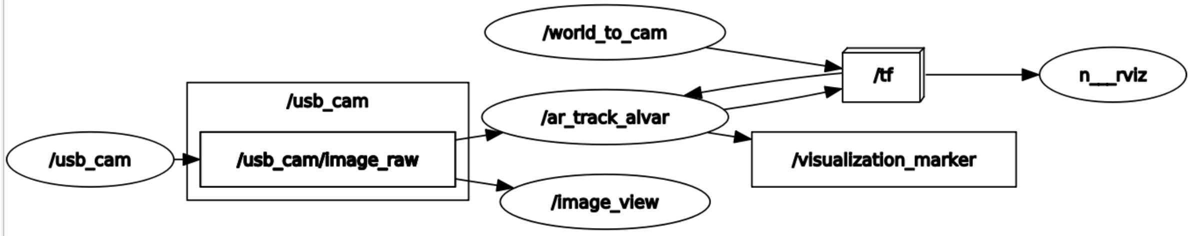
Topics posted:

话题名	数据类型
/visualization_marker	(visualization_msgs/Marker)
/ar_pose_marker	(ar_track_alvar/AlvarMarkers)

2.5 View the node diagram

Terminal input,

```
rqt_graph
```


/world_to_cam
/tf
n__rviz
/usb_cam
/ar_track_alvar
/usb_cam
/usb_cam/image_raw
/visualization_marker
/image_view