- # 34. AR QR code tracking

AR function package location: ~/ArTrack_ws/src/ar_track_alvar/ar_track_alvar

34.1. Overview

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:

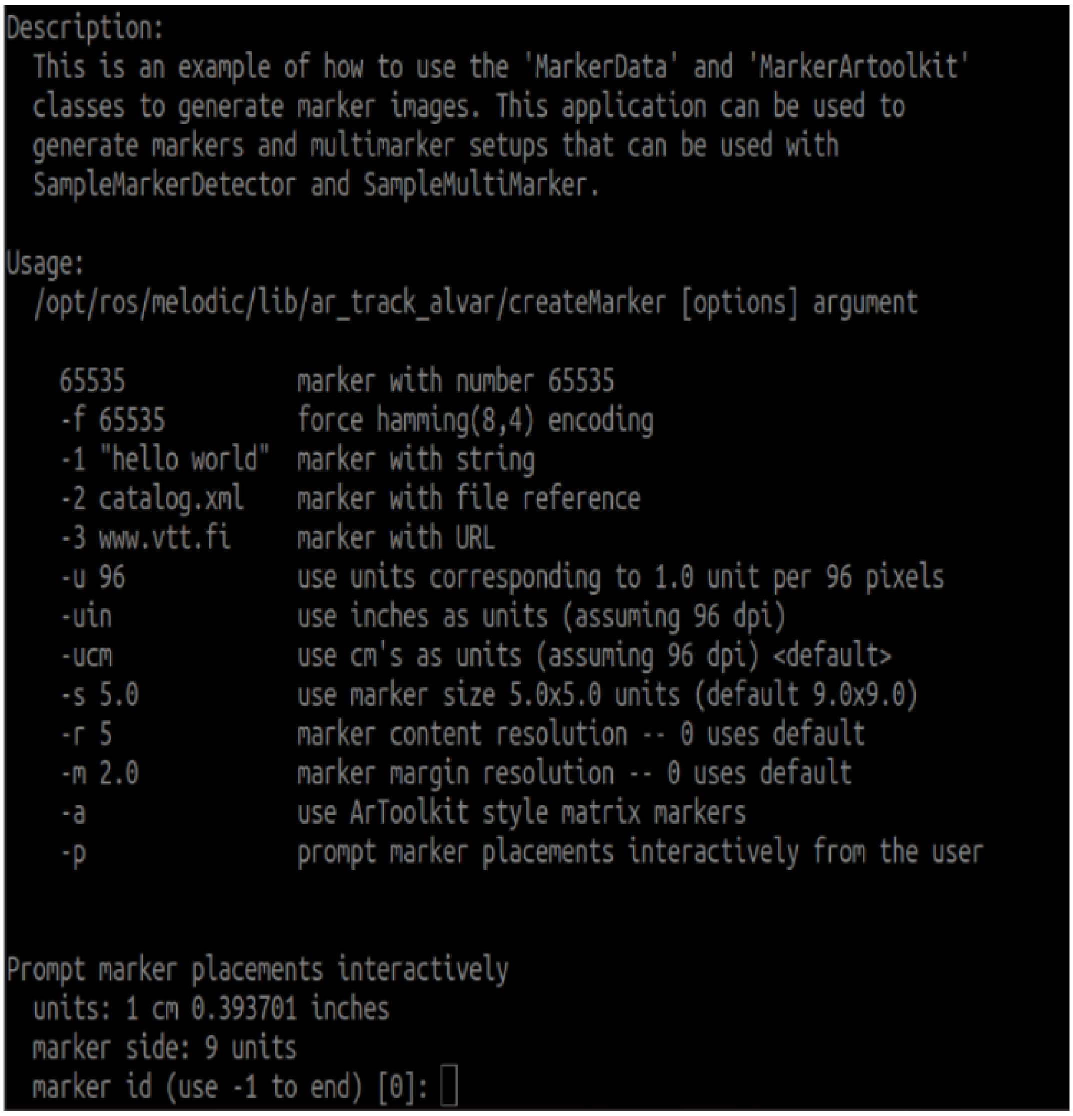
- Generate AR tags with different sizes, resolutions and data/ID encodings.
- Recognize and track poses of individual AR tags, optionally integrating kinect depth data (when kinect is available) for better pose estimation.
- Automatically calculate spatial relationships between tags in bundles using 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.

34.2. Create ARTag

- Continuously generate multiple tags on one image

```
roscore
rosrun ar_track_alvar createMarker
```



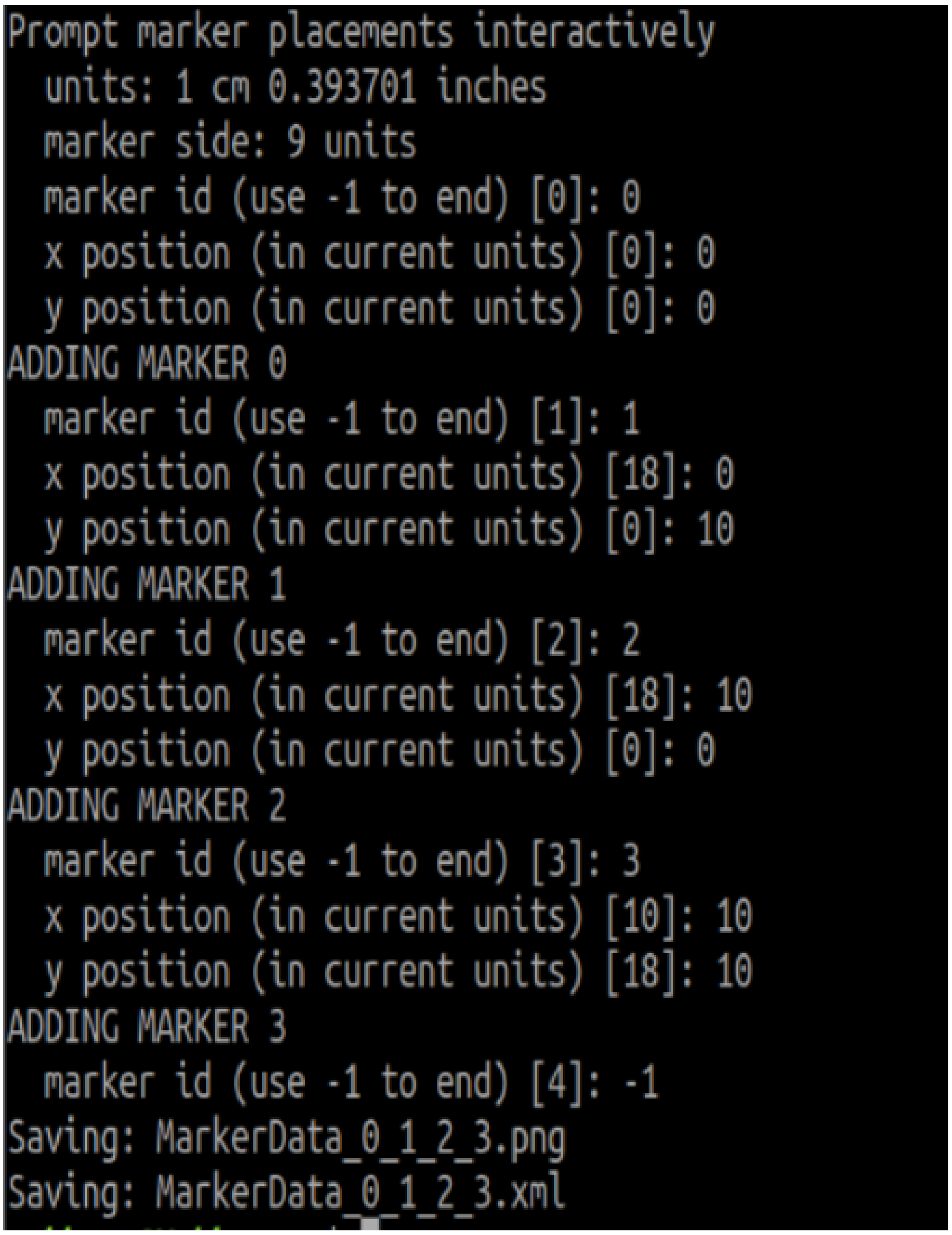
```
Description:
  This is an example of how to use the 'MarkerData' and 'MarkerArtoolkit'
  classes to generate marker images. This application can be used to
  generate markers and multimarker setups that can be used with
  SampleMarkerDetector and SampleMultiMarker.

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 digital pictures; 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 generated AR QR code is saved in the directory of the terminal running the command.

34.3. ARTag identification and tracking

Note: When starting the camera, you need to load the camera calibration file, otherwise it will not be recognized. The supporting virtual machine has been calibrated with the parameter file required when usb_cam is started. If there is no such parameter file, you need to calibrate it according to the following tutorial.

34.3.1. Camera internal parameter calibration

Install the camera calibration function package,

```
sudo apt install ros-noetic-camera-calibration -y
sudo apt install ros-noetic-usb-cam
```

Use usb-cam to drive the camera,

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

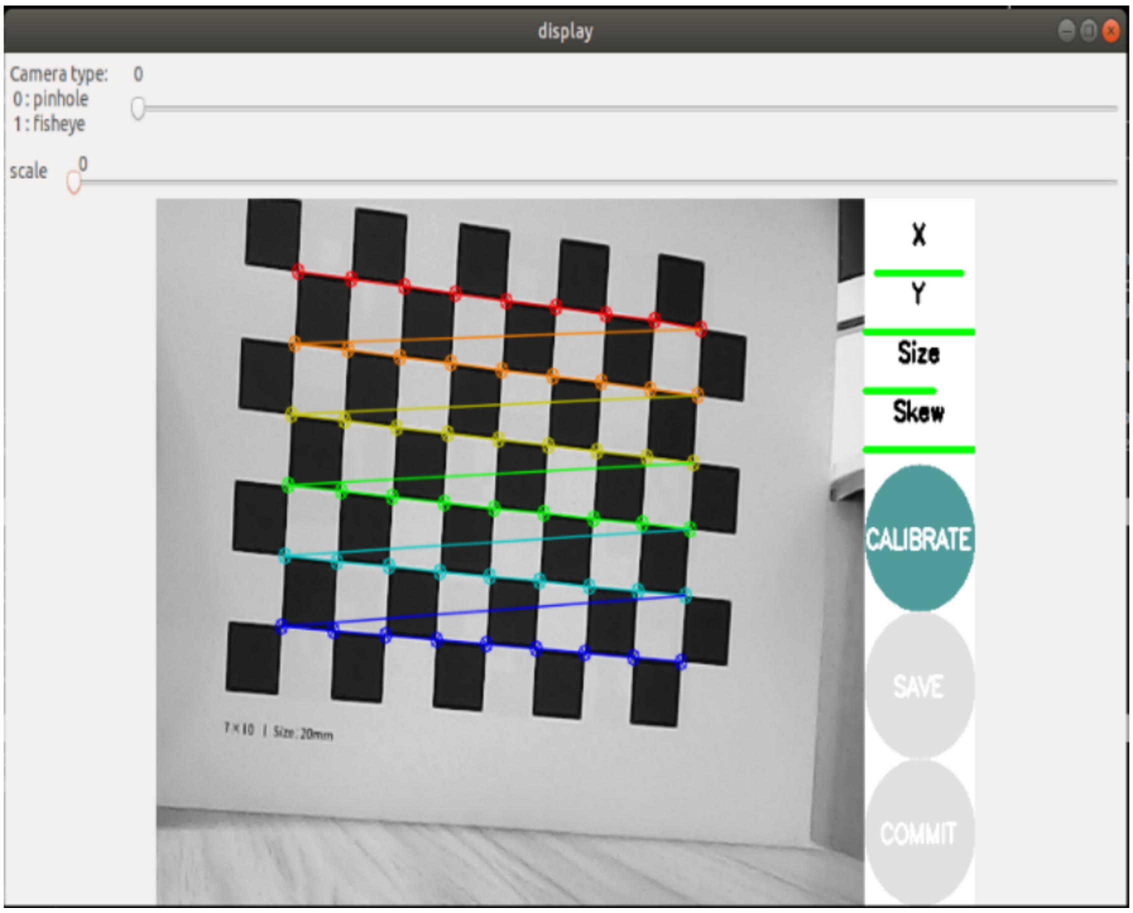
Start the calibrated node program,

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

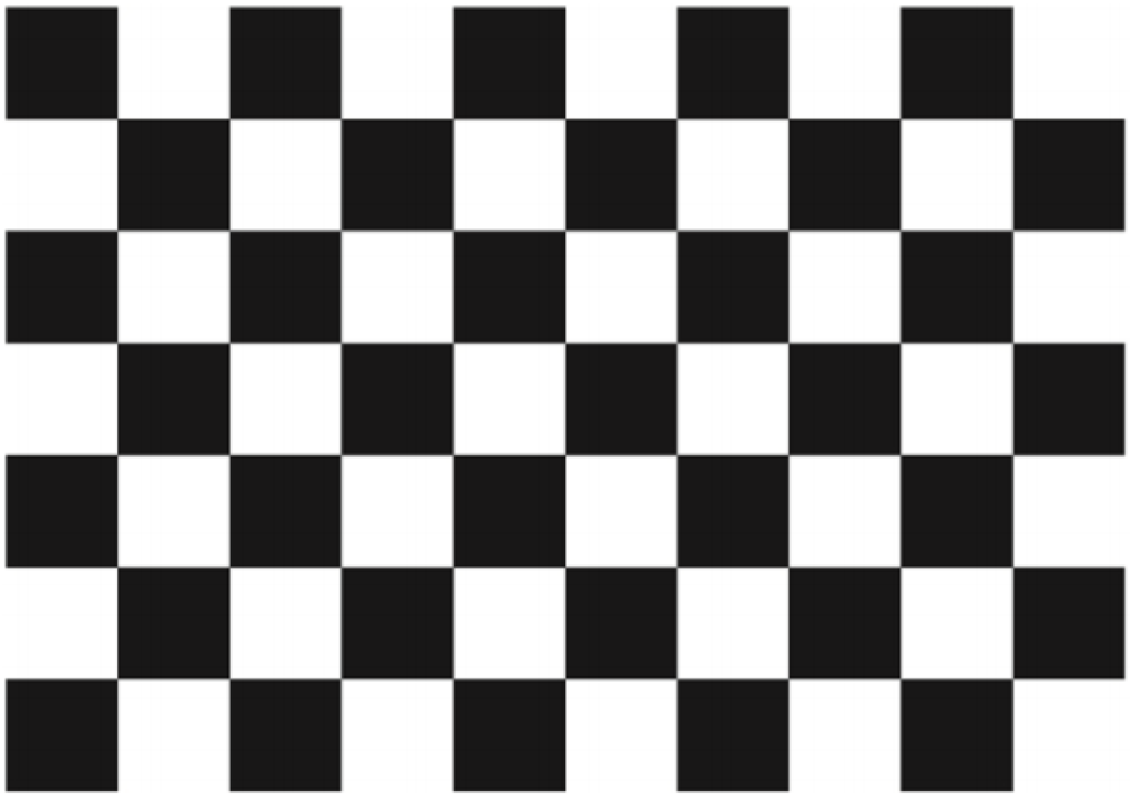
size: Calibrate the number of internal corner points of the checkerboard, for example, 9X6, with a total of six rows and nine columns of corner points.

square: The side length of the checkerboard, in meters.

image: Set the image topic published by the camera.



Place the prepared calibration checkerboard (7*10|size:20mm) in front of the camera.



X: The left and right movement of the checkerboard in the camera field of view

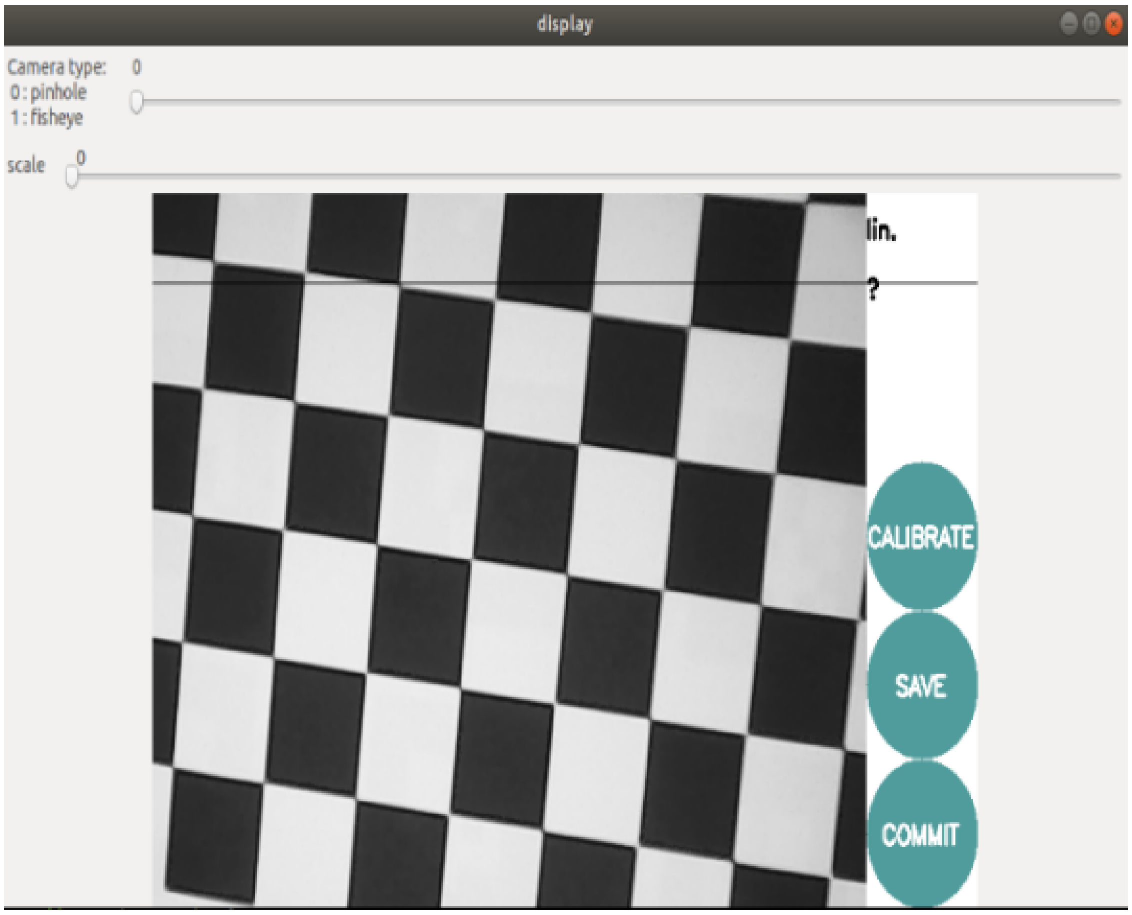
Y: The checkerboard moves up and down in the camera field of view

Size: the movement of the checkerboard back and forth in the camera field of view

Skew: The tilt and rotation of the checkerboard in the camera's field of view

After successful startup, place the checkerboard in the center of the screen and change to different positions. The system will identify it independently. The best situation is that the lines under [X], [Y], [Size], and [Skew] will first change from red to yellow and then to green as the data is collected, filling them as fully as possible.

Click [CALIBRATE] to calculate the internal parameters of the camera. The more pictures you have, the longer it will take. Just wait. (Sixty or seventy photos are enough, too many can easily cause jamming)



Click [SAVE] to save the calibration result. The calibration result is saved in the directory of the terminal running the calibration program. Use the following command to move it out.

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

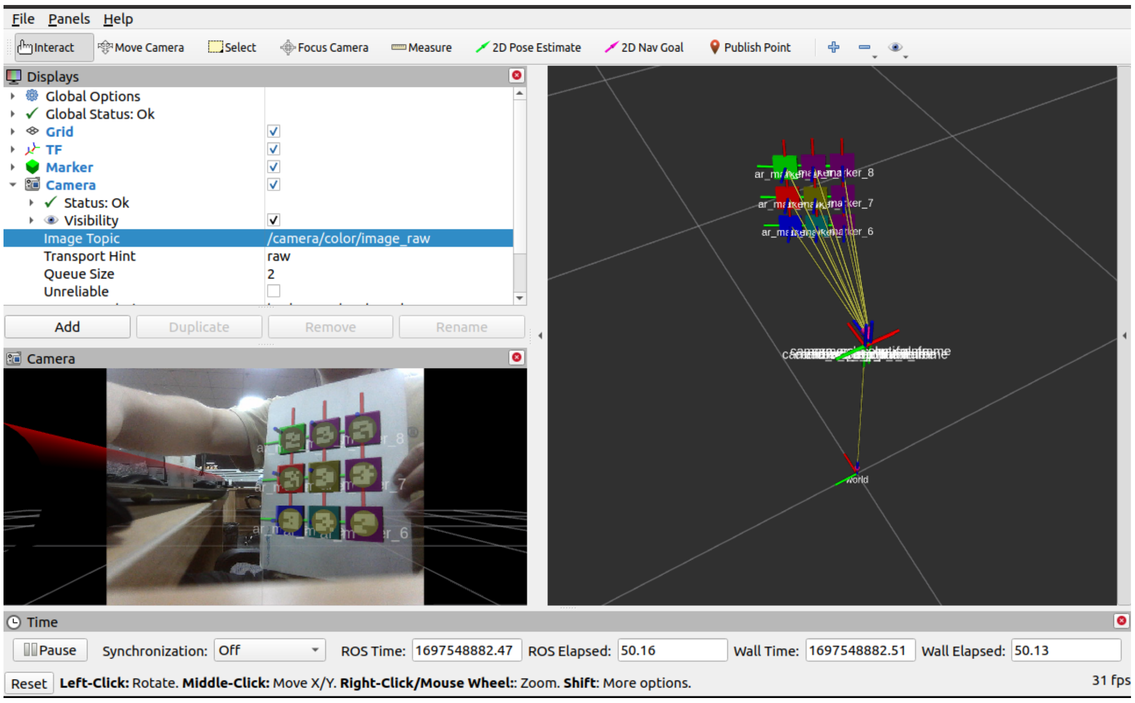
After decompression, there are the image just calibrated, an ost.txt file and an ost.yaml file. The yaml file is what we need, but it still needs to be modified before it can be used.

- Change ost.yaml to head_camera.yaml
- Move the file to the ~/.ros/camera_info folder

  ```
  cd ~/.ros
  sudo mkdir camera_info
  sudo cp ~/calibrationdata/head_camera.yaml ~/.ros/camera_info
  ```

34.3.2. Start ARTag identification and tracking

```
roslaunch astra_visual ar_track.launch
```



Global Options
Global Status: Ok
Grid ✓
TF ✓
Marker ✓
Camera ✓
Status: Ok
Visibility ✓
Image Topic /camera/color/image_raw
Transport Hint raw
Queue Size 2
Unreliable

Synchronization: Off ROS Time: 1697548882.47 ROS Elapsed: 50.16 Wall Time: 1697548882.51 Wall Elapsed: 50.13

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 31 fps

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

- Image_Topic: Obi Zhongguang's camera is [/camera/color/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.
- camera_link: camera coordinate system.

launch file parsing,

```
<launch>
     <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"/>
     <!-- Dabai_dcw2 camera configuration parameters -->
     <arg name="cam_image_topic" default="/camera/color/image_raw"/>
     <arg name="cam_info_topic" default="/camera/color/camera_info"/>
     <arg name="output_frame" default="/camera_link"/>
     <!--Camera startup node-->
     <include file="$(find orbbec_camera)/launch/orbbec_camera.launch"/>
     <node pkg="tf" type="static_transform_publisher" name="world_to_cam"
           args="0 0 0.5 0 0 0 world camera_link 10"/>
     <node name="ar_track_alvar" pkg="ar_track_alvar"
type="individualMarkersNoKinect" respawn="false" output="screen">
         <param name="marker_size" type="double" value="$(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)"/>
```

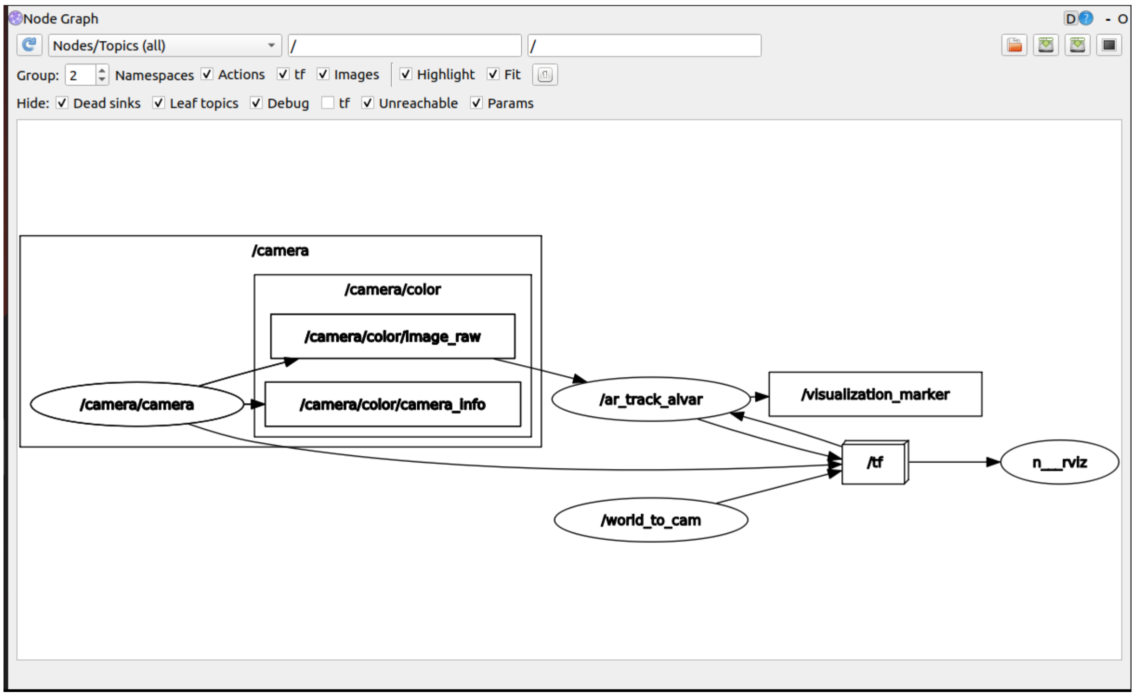
```
        <remap from="camera_image" to="$(arg cam_image_topic)"/>
        <remap from="camera_info" to="$(arg cam_info_topic)"/>
    </node>
    <group if="$(arg open_rviz)">
        <node pkg="rviz" type="rviz" name="rviz" args="-d $(find
astra_visual)/rviz/ar_track.rviz"/>
    </group>
</launch>
```

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.

34.3.3. View node graph

```
rqt_graph
```


Node Graph

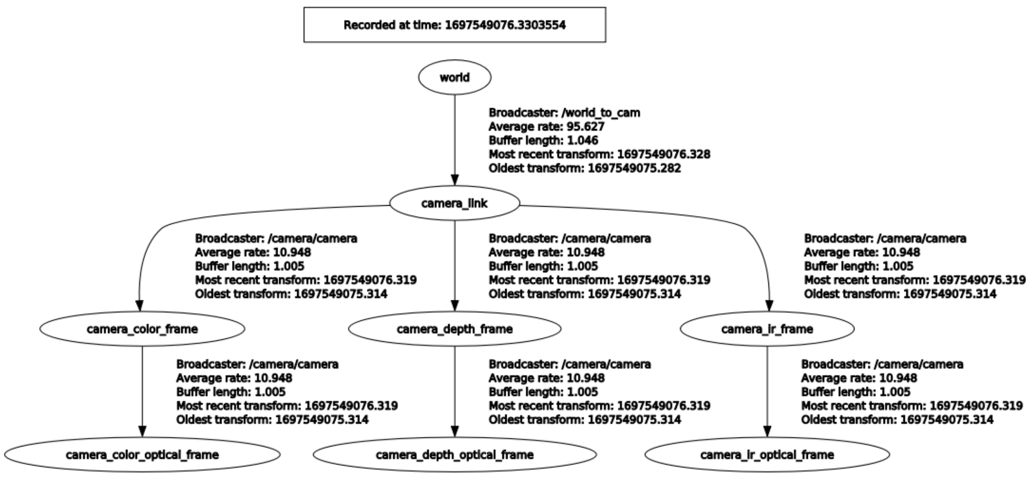
Nodes/Topics (all) / /

Group: 2 Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit

Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug ☐ tf ✓ Unreachable ✓ Params

34.3.4. View TF tree

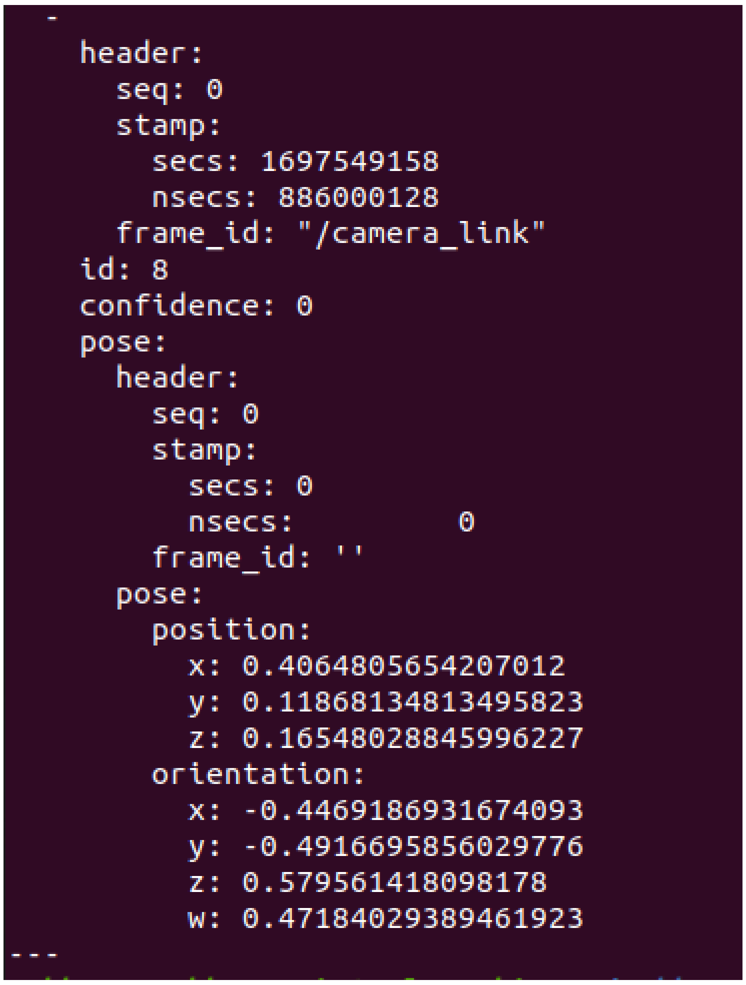
```
rosrun rqt_tf_tree rqt_tf_tree
```


Recorded at time: 1697549076.3303554
world
Broadcaster: /world_to_cam
Average rate: 95.627
Buffer length: 1.046
Most recent transform: 1697549076.328
Oldest transform: 1697549075.282
camera_link
Broadcaster: /camera/camera
Average rate: 10.948
Buffer length: 1.005
Most recent transform: 1697549076.319
Oldest transform: 1697549075.314
Broadcaster: /camera/camera
Average rate: 10.948
Buffer length: 1.005
Most recent transform: 1697549076.319
Oldest transform: 1697549075.314
Broadcaster: /camera/camera
Average rate: 10.948
Buffer length: 1.005
Most recent transform: 1697549076.319
Oldest transform: 1697549075.314
camera_color_frame
camera_depth_frame
camera_ir_frame
Broadcaster: /camera/camera
Average rate: 10.948
Buffer length: 1.005
Most recent transform: 1697549076.319
Oldest transform: 1697549075.314
Broadcaster: /camera/camera
Average rate: 10.948
Buffer length: 1.005
Most recent transform: 1697549076.319
Oldest transform: 1697549075.314
Broadcaster: /camera/camera
Average rate: 10.948
Buffer length: 1.005
Most recent transform: 1697549076.319
Oldest transform: 1697549075.314
camera_color_optical_frame
camera_depth_optical_frame
camera_ir_optical_frame

34.3.5. View the output topic information

```
rostopic echo /ar_pose_marker
```

Displayed as follows:



```
 -
   header:
     seq: 0
     stamp:
       secs: 1697549158
       nsecs: 886000128
     frame_id: "/camera_link"
   id: 8
   confidence: 0
   pose:
     header:
       seq: 0
       stamp:
         secs: 0
         nsecs:         0
       frame_id: ''
     pose:
       position:
         x: 0.4064805654207012
         y: 0.11868134813495823
         z: 0.16548028845996227
       orientation:
         x: -0.4469186931674093
         y: -0.4916695856029776
         z: 0.579561418098178
         w: 0.47184029389461923
---
```

- frame_id: The coordinate system name of the camera

- id: The number recognized is 8
- pose: the pose of the QR code
- position: the position of the QR code coordinate system relative to the camera coordinate system
- orientation: the orientation of the QR code coordinate system relative to the camera coordinate system