

# 1.HD camera calibration

---

## 1.HD camera calibration

### 1.1. Preparation before calibration

### 1.2. Calibration

Wiki: [http://wiki.ros.org/camera\\_calibration](http://wiki.ros.org/camera_calibration)

Ordinary camera: [https://github.com/bosch-ros-pkg/usb\\_cam.git](https://github.com/bosch-ros-pkg/usb_cam.git)

Monocular teaching: [http://wiki.ros.org/camera\\_calibration/Tutorials/MonocularCalibration](http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration)

Due to some internal and external reasons of the camera, the image will be greatly distorted, mainly radial deformation and tangential deformation, causing the straight line to become curved. The farther the pixel is from the center of the image, the more serious the distortion will be. In order to avoid errors caused by data sources, the parameters of the camera need to be calibrated. Calibration essentially uses a known and determined spatial relationship (calibration plate) to reversely deduce the inherent and real parameters of the camera (internal parameters) by analyzing the pixels of the photographed pictures.

## 1.1. Preparation before calibration

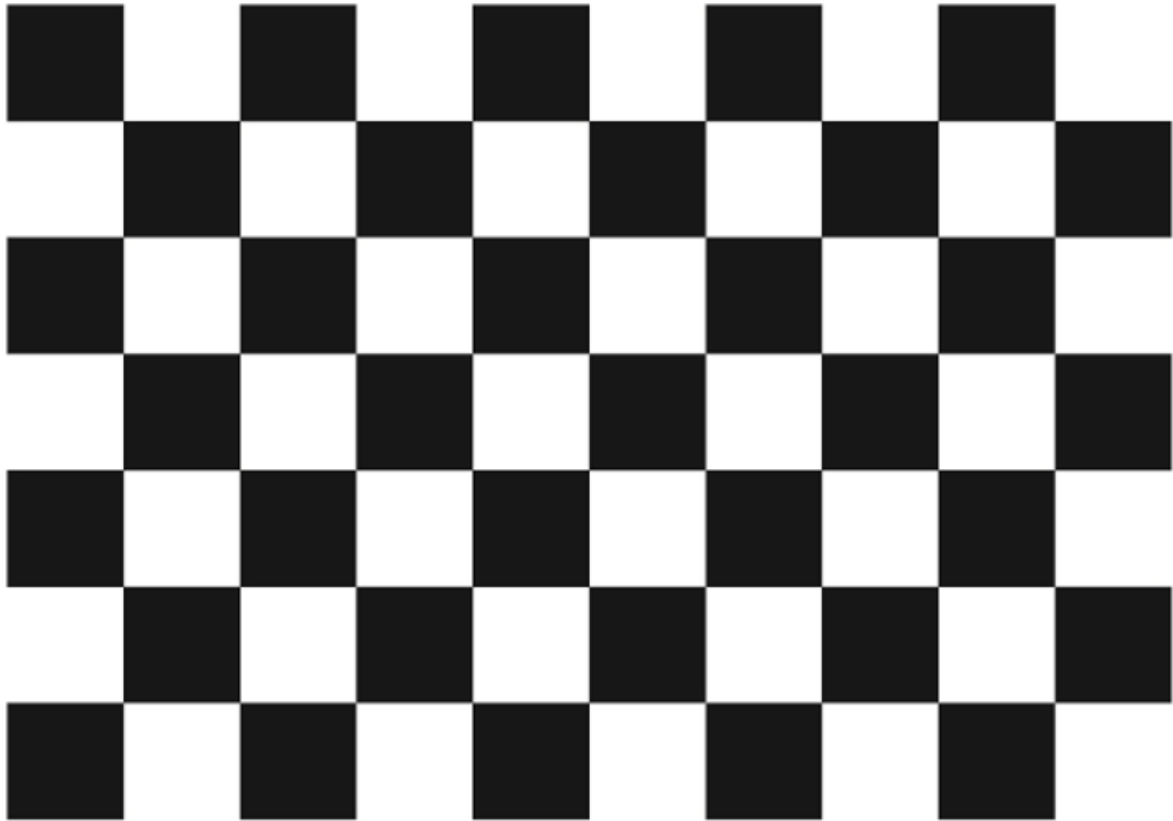
- A large [chessboard] of known dimensions([http://wiki.ros.org/camera\\_calibration/Tutorials/MonocularCalibration?action=AttachFile&do=view&target=check-108.pdf](http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration?action=AttachFile&do=view&target=check-108.pdf)). This tutorial uses a 9x6 checkerboard and a 20mm square, which needs to be flattened during calibration.

**The calibration uses the internal vertices of the checkerboard, so a "10x7" checkerboard uses the internal vertex parameters "9x6", as shown in the example below.**

Any calibration board can be used, as long as the parameters are changed.

- An open area without obstacles and calibration board patterns
- Monocular camera for publishing images via ROS

Checkerboard (calibration board)



7×10 | Size: 20mm

## 1.2. Calibration

Start camera

**jetson motherboard/Raspberry Pi 4B**

```
roscore  
roslaunch uvc_camera uvc_camera_node device:=/dev/video0
```

**Raspberry Pi 5**

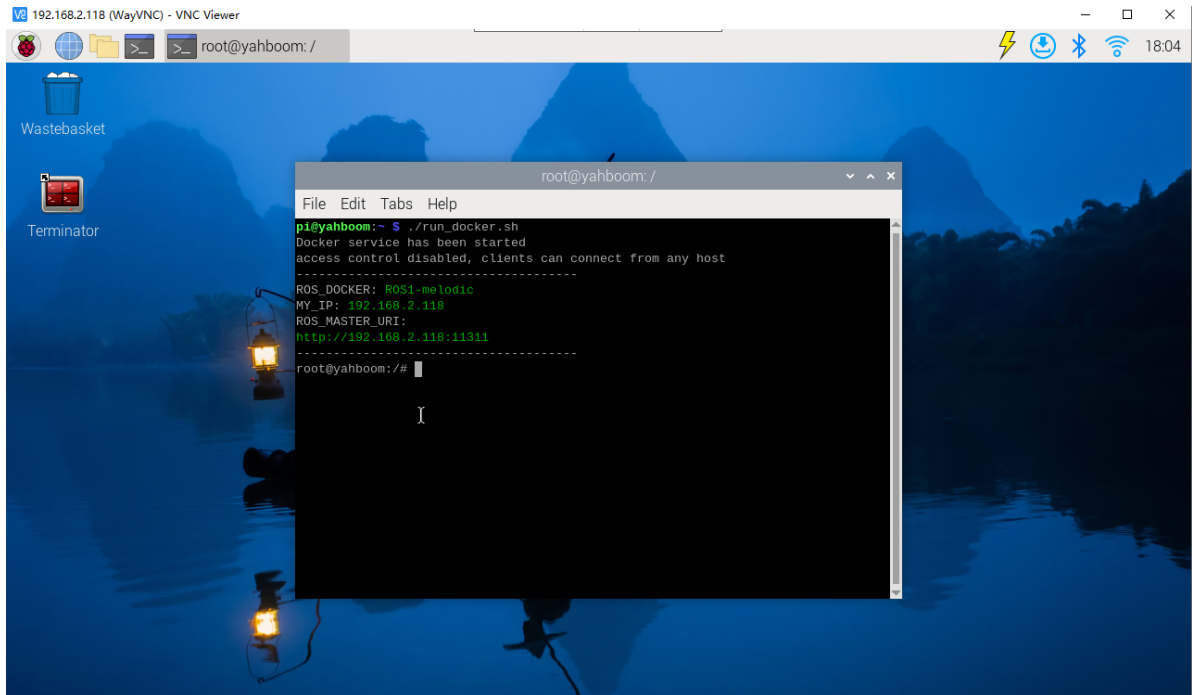
**Before running, please confirm that the large program has been permanently closed**

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
```



```
roscore
```

Enter the same docker from multiple terminals

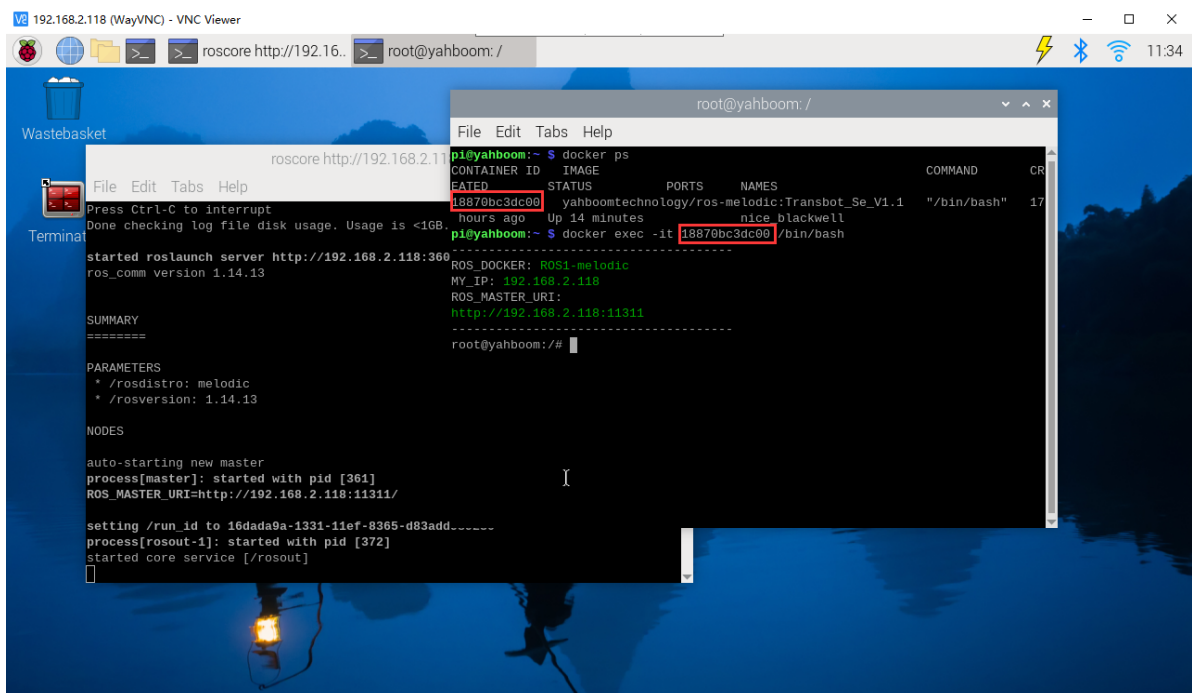
Keep the program of the previous docker terminal running and open a new terminal

Enter the following command

```
docker ps
```

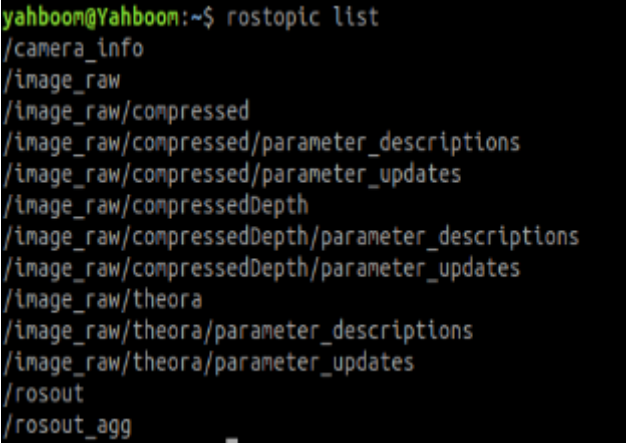
Enter the same docker and use the following 18870bc3dc00 to modify the ID displayed on the actual terminal.

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



```
roslaunch uvc_camera uvc_camera_node device:=/dev/video0
```

View Image Topics

A terminal window with a black background and green text. The prompt is 'yahboom@Yahboom:~\$'. The command 'rostopic list' has been executed, and the output lists several ROS topics: /camera\_info, /image\_raw, /image\_raw/compressed, /image\_raw/compressed/parameter\_descriptions, /image\_raw/compressed/parameter\_updates, /image\_raw/compressedDepth, /image\_raw/compressedDepth/parameter\_descriptions, /image\_raw/compressedDepth/parameter\_updates, /image\_raw/theora, /image\_raw/theora/parameter\_descriptions, /image\_raw/theora/parameter\_updates, /rosout, and /rosout\_agg.

```
yahboom@Yahboom:~$ rostopic list
/camera_info
/image_raw
/image_raw/compressed
/image_raw/compressed/parameter_descriptions
/image_raw/compressed/parameter_updates
/image_raw/compressedDepth
/image_raw/compressedDepth/parameter_descriptions
/image_raw/compressedDepth/parameter_updates
/image_raw/theora
/image_raw/theora/parameter_descriptions
/image_raw/theora/parameter_updates
/rosout
/rosout_agg
```

Start calibration node

**jetson motherboard/Raspberry Pi 4B**

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

**Raspberry Pi 5**

Enter the same docker from multiple terminals

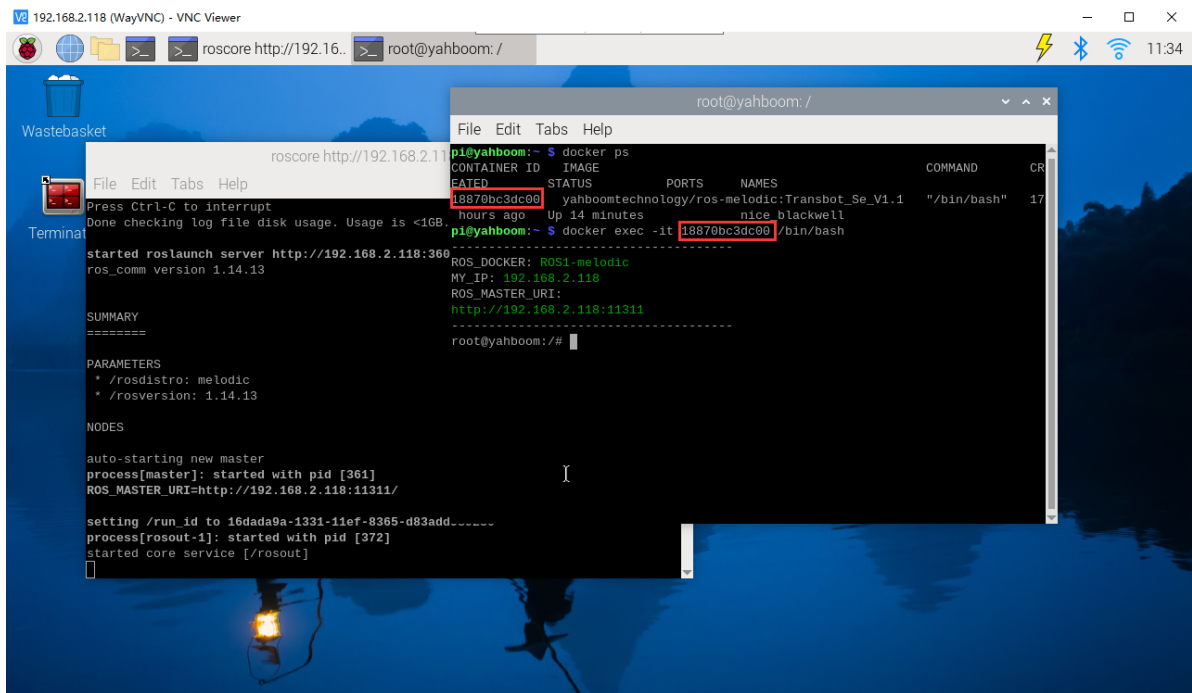
Keep the program of the previous docker terminal running and open a new terminal

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
```

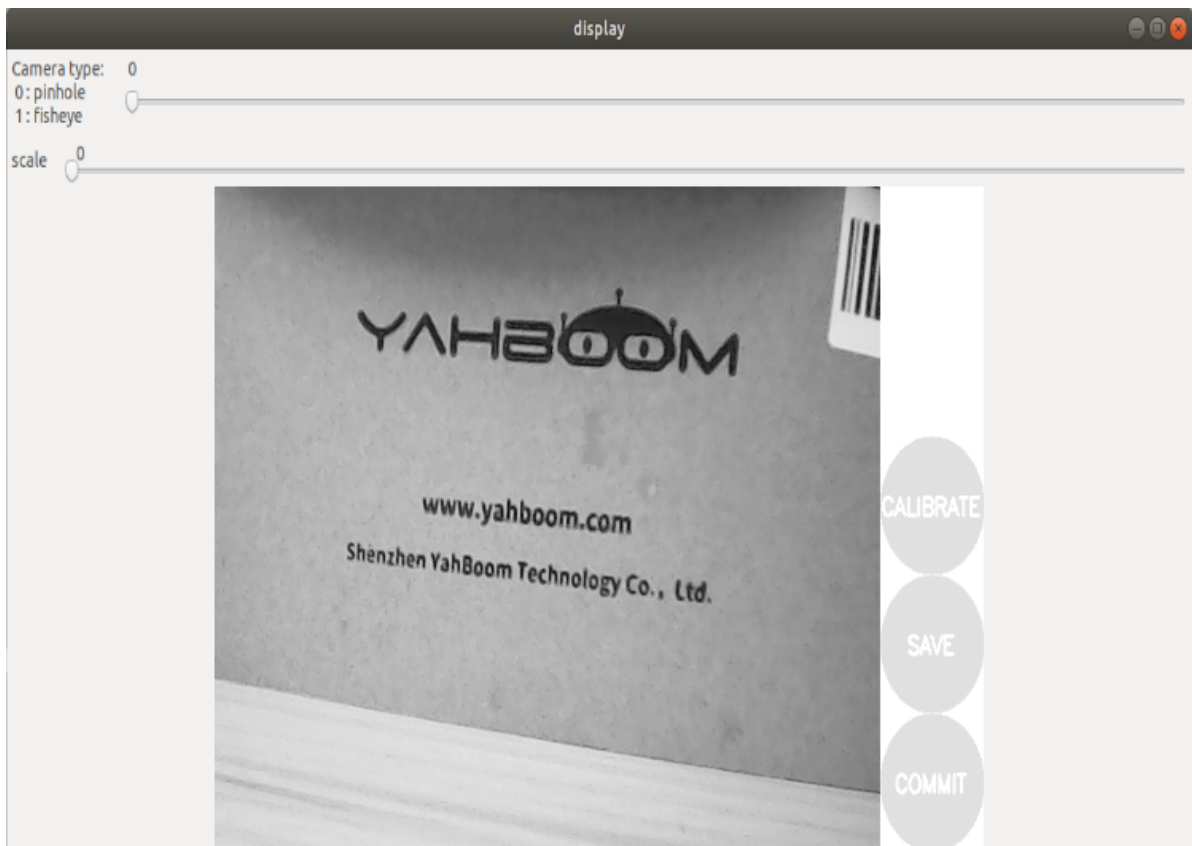


```
roslaunch camera_calibration cameracalibrator.py image:=/image_raw camera:=/ --size
9x6 --square 0.02
```

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 and camera: Set the image topic published by the camera.



Calibration interface

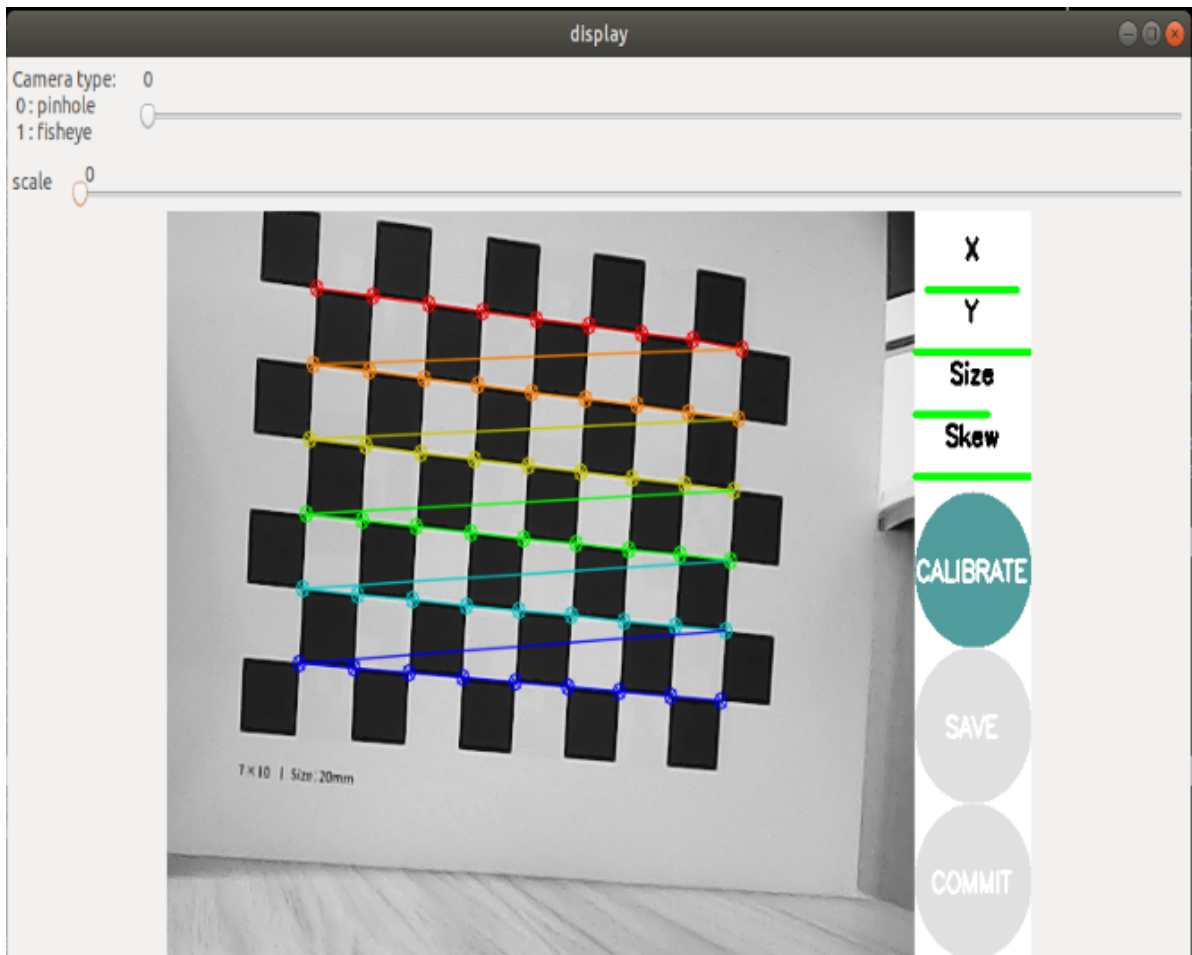
X: The left and right movement of the checkerboard in the camera field of view

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

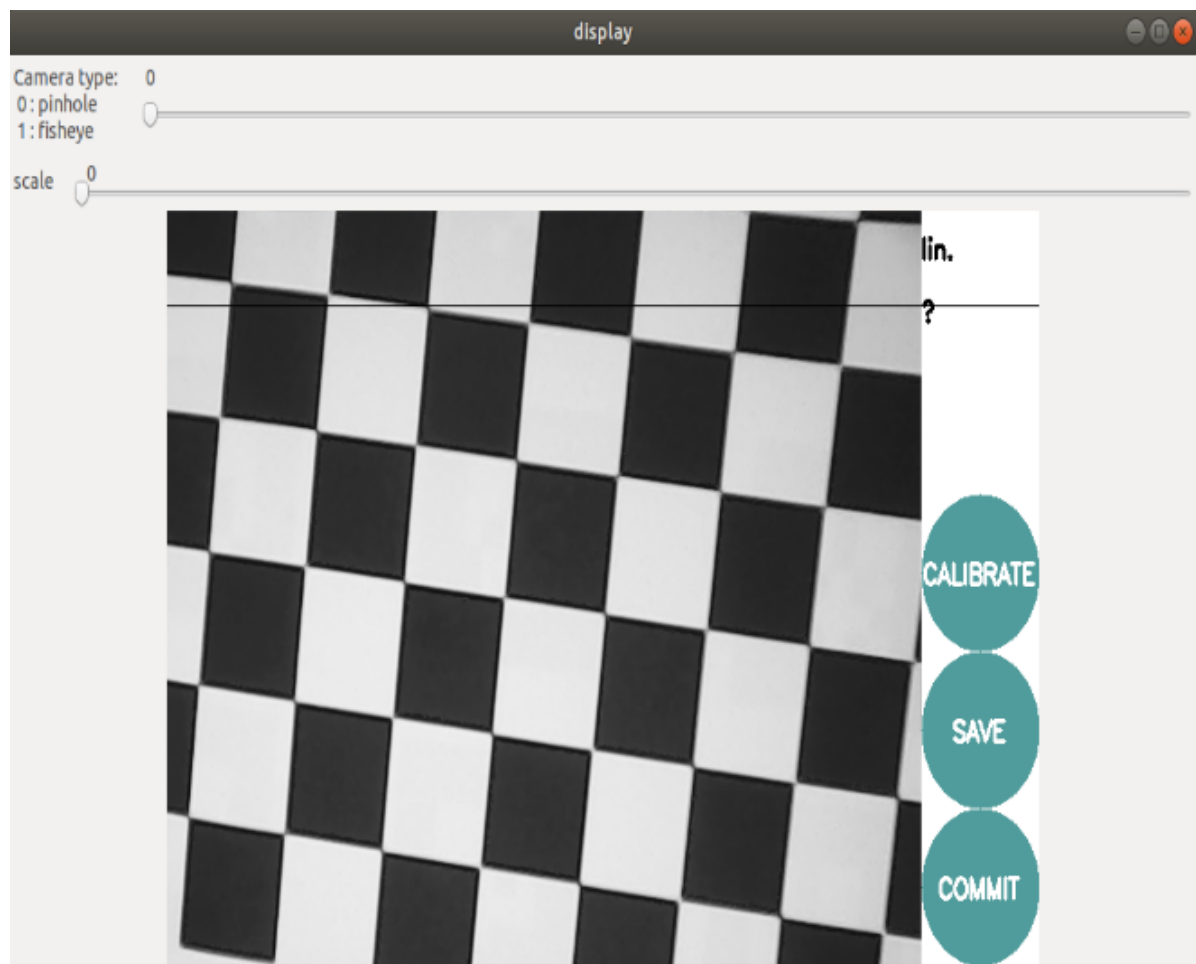
Size: the movement of the checkerboard back and forth in the camera field of view

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 pictures are enough, too many will easily cause jamming)



Click [SAVE] to save the non-calibration result. The bottom line appears. Click [COMMIT] to exit.

```

**** Calibrating ****
*** Added sample 46, p_x = 0.760, p_y = 0.692, p_size = 0.601, skew = 0.209
D = [0.15472379170842887, -0.8099148429709959, 0.001638104297210457, -0.004547715577735416, 0.0]
K = [908.7489797733236, 0.0, 206.95349685217326, 0.0, 909.6731941878273, 301.76750022317515, 0.0, 0.0, 1.0]
R = [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P = [903.021484375, 0.0, 203.24846800838714, 0.0, 0.0, 916.7886962890625, 302.28809574724073, 0.0, 0.0, 0.0, 1.0, 0.0]
None
# oST version 5.0 parameters

[image]

width
640

height
480

[narrow_stereo]

camera matrix
908.748980 0.000000 206.953497
0.000000 909.673194 301.767500
0.000000 0.000000 1.000000

distortion
0.154724 -0.809915 0.001638 -0.004548 0.000000

rectification
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000

projection
903.021484 0.000000 203.248468 0.000000
0.000000 916.788696 302.288096 0.000000
0.000000 0.000000 1.000000 0.000000

('Wrote calibration data to', '/tmp/calibrationdata.tar.gz')

```

After the calibration is completed, the calibration results are stored in [/tmp/calibrationdata.tar.gz] and can be moved out to view the contents.

```
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 camera.yaml
- Change the name after camera\_name: to camera
- Move the file to the ~/.ros/camera\_info folder