9. ROS+Opencv application

9.1. Overview

wiki: http://wiki.ros.org/opencv_apps

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

Function package: ~/software/transbot_library/src/opencv_apps

Most of the code was originally taken from https://github.com/Itseez/opencv/tree/master/samples/cpp

The opencv_apps program provides various nodes that internally run the functions of opencv and publish the results to a ROS topic. When using the opencv_apps program, you only need to run a launch file according to your own business needs, so you don't have to write program codes for these functions.

ROS Wiki has related node analysis, topic subscription and topic publishing of the corresponding node, introduction of related parameters, etc. See the ROS WiKi for details.

Contents

9.2. Use

9.2.1. Start

Step 1: Start the camera

jetson motherboard/Raspberry Pi 4B

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

Raspberry Pi 5

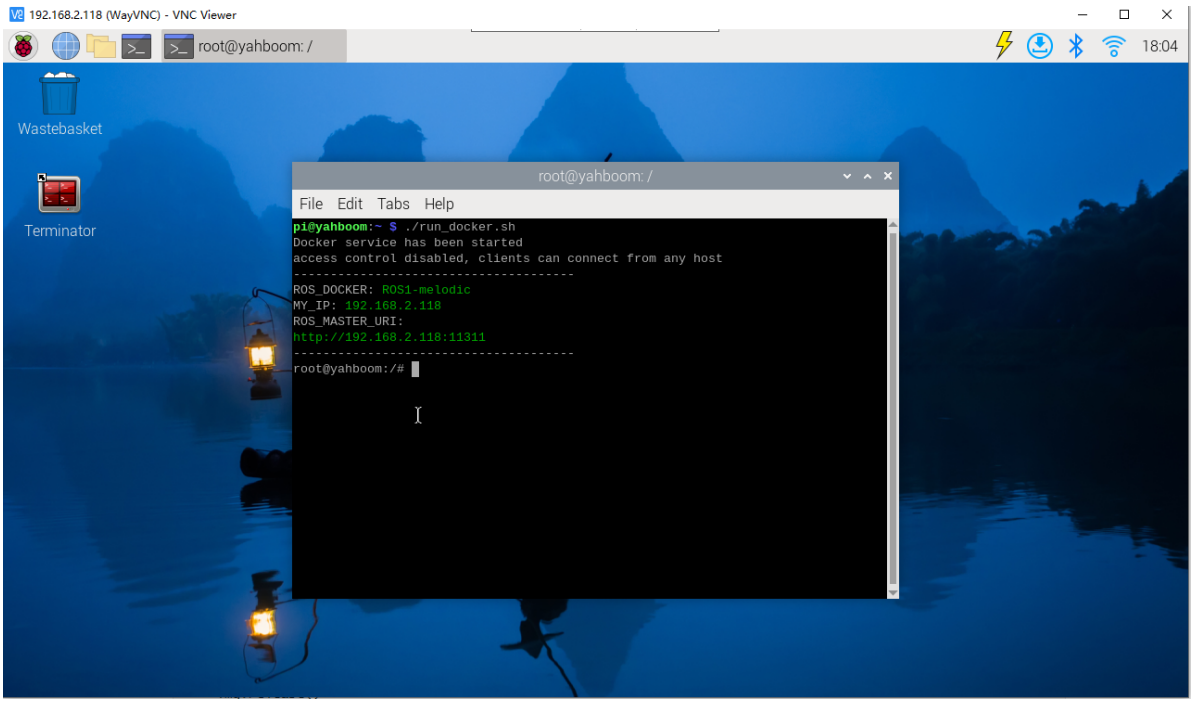
Before running, please confirm that the large program has been permanently closed

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
```




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

[camDevice]: The high frame rate camera is [USBCam];

If the web page cannot be viewed, check whether there is no [web_video_server] node. If not, run the following command

jetson motherboard/Raspberry Pi 4B

```
rosrun web_video_server web_video_server
```

Raspberry Pi 5

Enter the same docker from multiple terminals

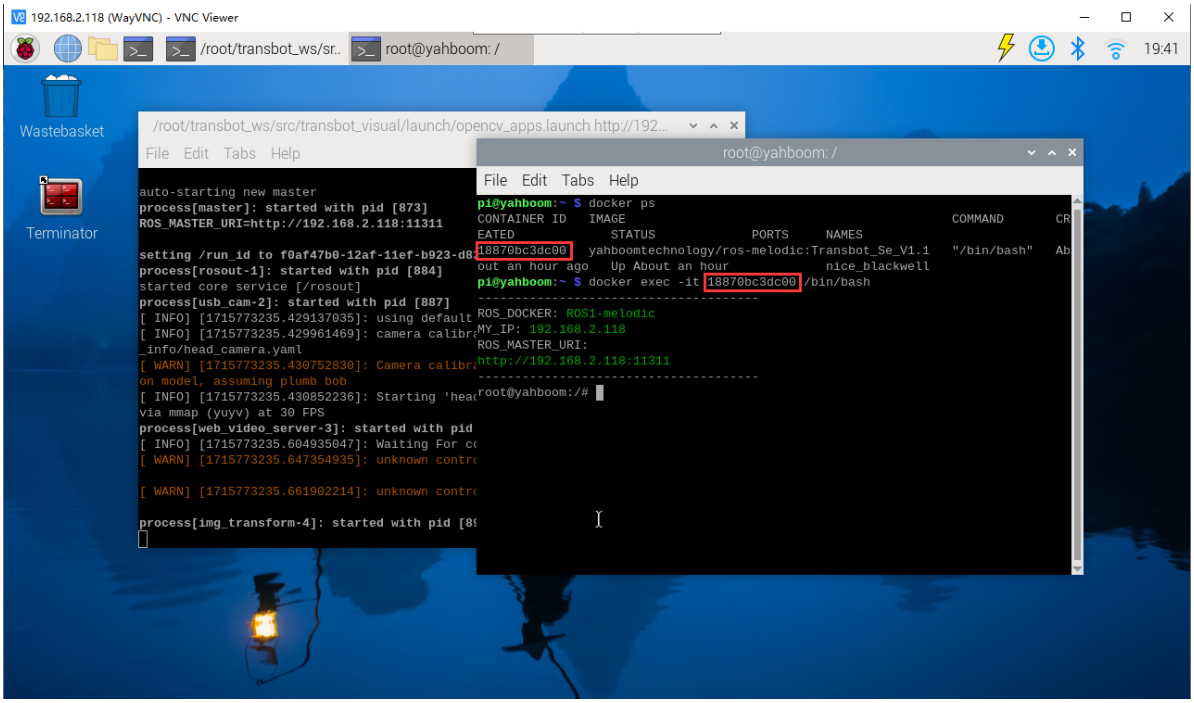
Keep the program of the previous docker terminal running and open a new terminal

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
```



Step 2: Start the function of Opencv_apps

jetson motherboard/Raspberry Pi 4B

```
roslaunch opencv_apps face_recognition.launch # Face recognition
roslaunch opencv_apps corner_harris.launch # harris corner detection
roslaunch opencv_apps camshift.launch # Target tracking algorithm
roslaunch opencv_apps contour_moments.launch # Contour moments
roslaunch opencv_apps convex_hull.launch # Polygon outline
roslaunch opencv_apps discrete_fourier_transform.launch # Discrete Fourier
transform algorithm
roslaunch opencv_apps edge_detection.launch # Edge detection algorithm
roslaunch opencv_apps face_detection.launch # Face detection algorithm
roslaunch opencv_apps fback_flow.launch # Optical flow detection algorithm
roslaunch opencv_apps find_contours.launch # Contour detection
roslaunch opencv_apps general_contours.launch # General contour detection
roslaunch opencv_apps goodfeature_track.launch # Feature point tracking
roslaunch opencv_apps hls_color_filter.launch # HLS color filtering
roslaunch opencv_apps hough_circles.launch # Hough circle detection
roslaunch opencv_apps hough_lines.launch # Hough line detection
roslaunch opencv_apps hsv_color_filter.launch # HSV color filtering
roslaunch opencv_apps lk_flow.launch # LK optical flow algorithm
roslaunch opencv_apps people_detect.launch # Human detection algorithm
roslaunch opencv_apps phase_corr.launch # Phase-related displacement detection
roslaunch opencv_apps pyramids.launch # Image pyramid sampling algorithm
roslaunch opencv_apps rgb_color_filter.launch # RGB color filtering
roslaunch opencv_apps segment_objects.launch # Clear background detection
algorithm
roslaunch opencv_apps simple_flow.launch # Simplified optical flow algorithm
roslaunch opencv_apps smoothing.launch # Simple filter
roslaunch opencv_apps threshold.launch # Threshold image processing
roslaunch opencv_apps watershed_segmentation.launch # Watershed segmentation
algorithm
```

Raspberry Pi 5

Enter the same docker from multiple terminals

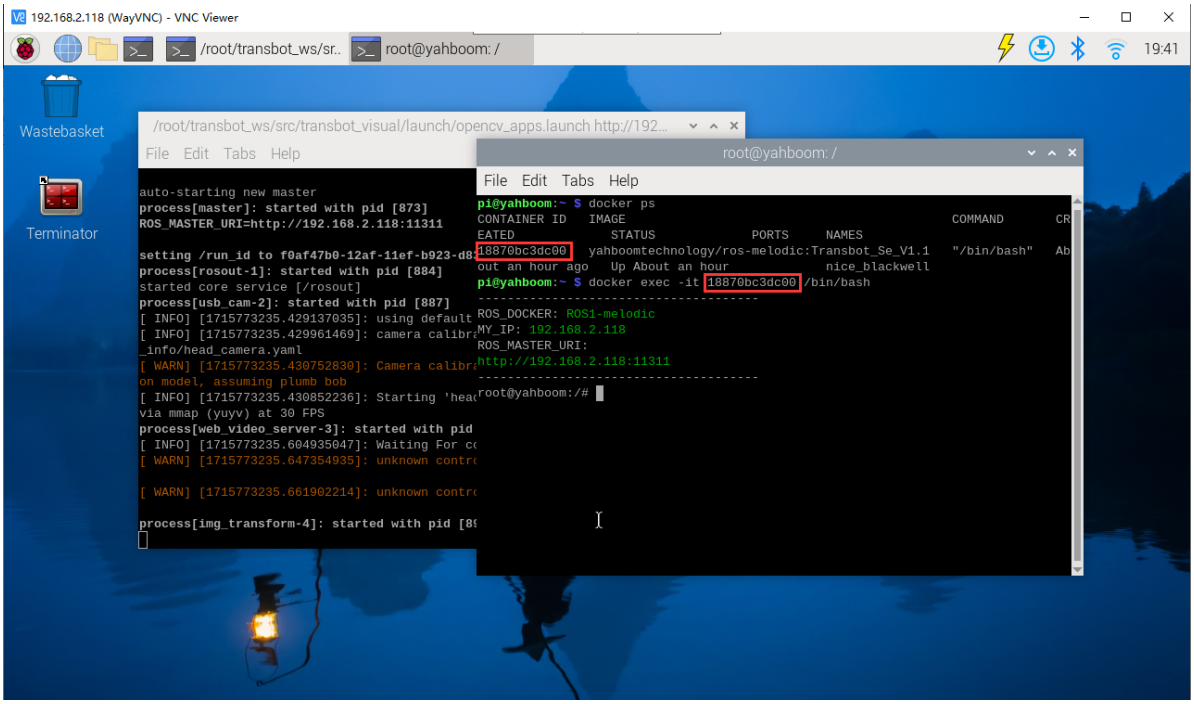
Keep the program of the previous docker terminal running and open a new terminal

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 opencv_apps face_recognition.launch # Face recognition
roslaunch opencv_apps corner_harris.launch # harris corner detection
roslaunch opencv_apps camshift.launch # Target tracking algorithm
roslaunch opencv_apps contour_moments.launch # Contour moments
roslaunch opencv_apps convex_hull.launch # Polygon outline
roslaunch opencv_apps discrete_fourier_transform.launch # Discrete Fourier
transform algorithm
roslaunch opencv_apps edge_detection.launch # Edge detection algorithm
roslaunch opencv_apps face_detection.launch # Face detection algorithm
roslaunch opencv_apps fback_flow.launch # Optical flow detection algorithm
roslaunch opencv_apps find_contours.launch # Contour detection
roslaunch opencv_apps general_contours.launch # General contour detection
roslaunch opencv_apps goodfeature_track.launch # Feature point tracking
roslaunch opencv_apps hls_color_filter.launch # HLS color filtering
roslaunch opencv_apps hough_circles.launch # Hough circle detection
roslaunch opencv_apps hough_lines.launch # Hough line detection
roslaunch opencv_apps hsv_color_filter.launch # HSV color filtering
roslaunch opencv_apps lk_flow.launch # LK optical flow algorithm
roslaunch opencv_apps people_detect.launch # Human detection algorithm
roslaunch opencv_apps phase_corr.launch # Phase-related displacement detection
roslaunch opencv_apps pyramids.launch # Image pyramid sampling algorithm
roslaunch opencv_apps rgb_color_filter.launch # RGB color filtering
```

```
roslaunch opencv_apps segment_objects.launch # Clear background detection
algorithm
roslaunch opencv_apps simple_flow.launch # Simplified optical flow algorithm
roslaunch opencv_apps smoothing.launch # Simple filter
roslaunch opencv_apps threshold.launch # Threshold image processing
roslaunch opencv_apps watershed_segmentation.launch # Watershed segmentation
algorithm
```

Almost every functional case will have a parameter [debug_view], Boolean type, whether to use Opencv to display pictures, which is displayed by default.

If no display is required, set it to [False], for example

```
roslaunch opencv_apps contour_moments.launch debug_view:=False
```

However, after starting in this way, some cases cannot be displayed in other ways, because in the source code, some [debug_view] is set to [False], which will turn off image processing.

9.2.2. Display method

- rqt_image_view

Enter the following command to select the corresponding topic

jetson motherboard/Raspberry Pi 4B

```
rqt_image_view
```

Raspberry Pi 5

Enter the same docker from multiple terminals

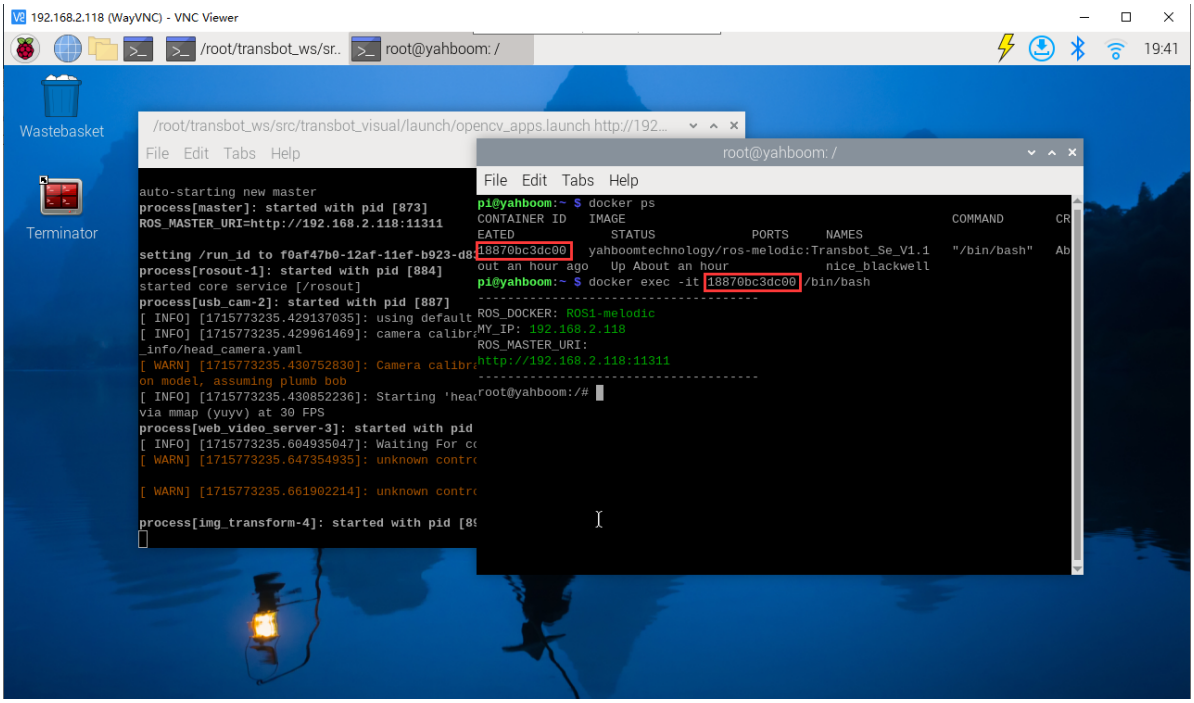
Keep the program of the previous docker terminal running and open a new terminal

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
```




```
rqt_image_view
```

- opencv

The system displays it by default, no need to do anything.

- Web viewing

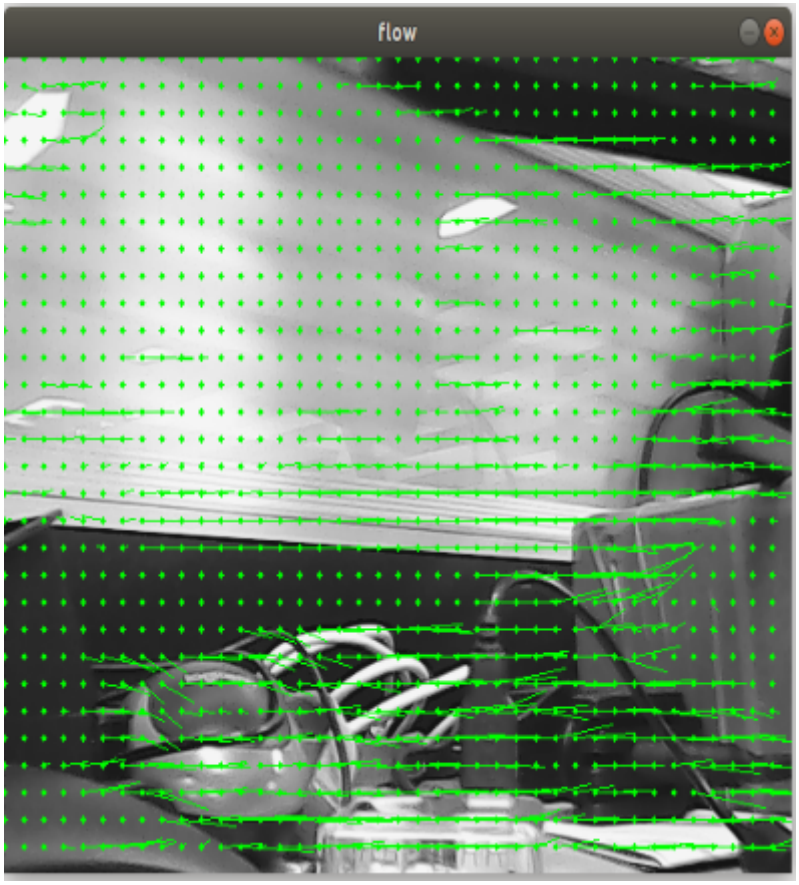
(Same as under LAN) Enter IP+port in the browser, for example:

```
192.168.2.102:8080
```

9.2.3. Effect display

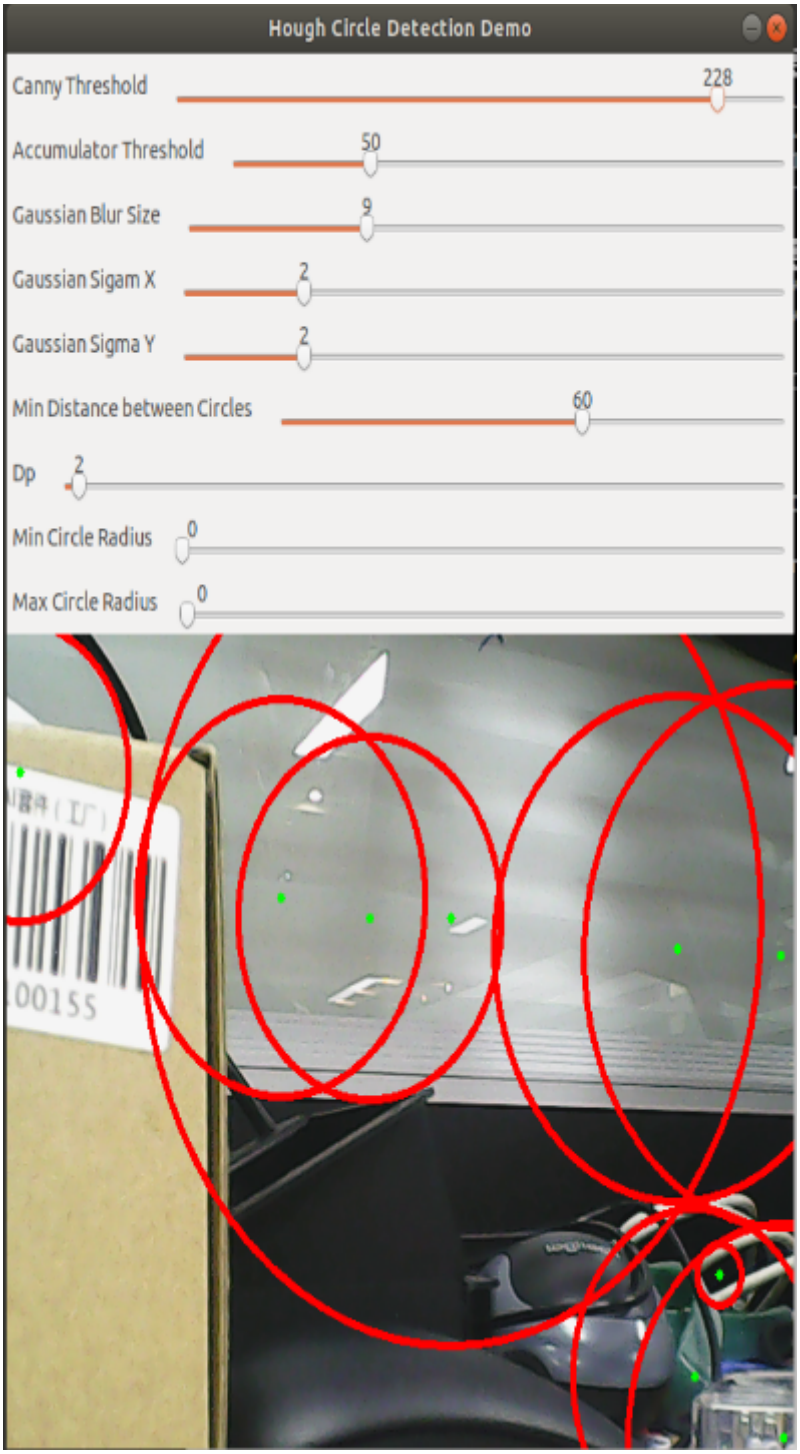
- Optical flow detection algorithm

Move the screen and observe the phenomenon.



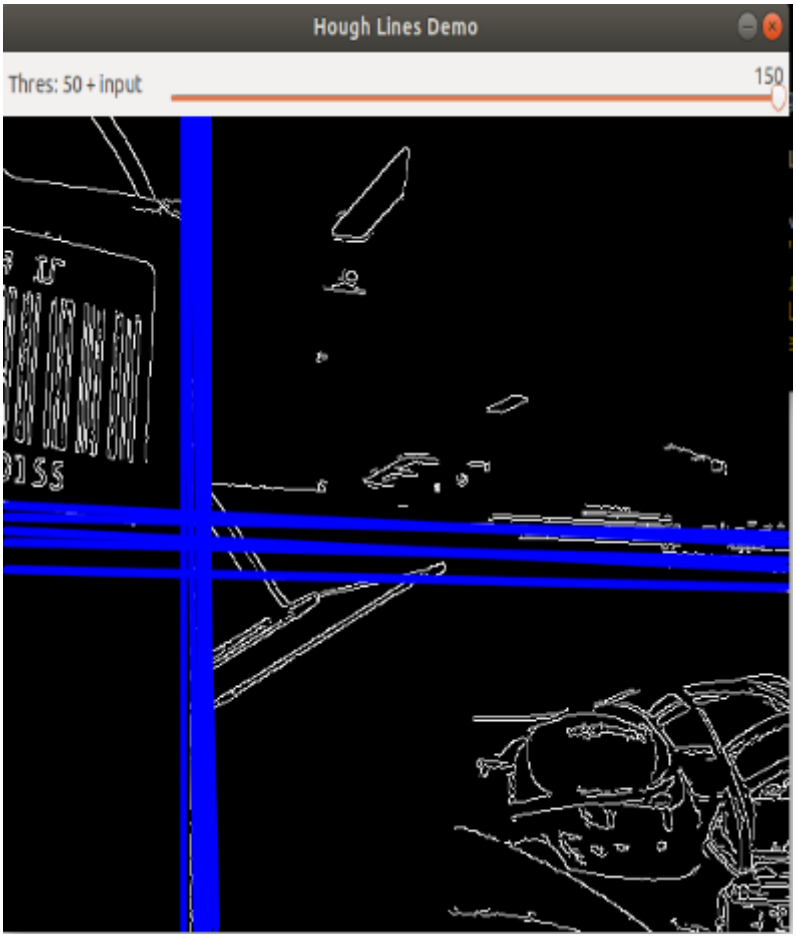
- Feature point tracking

- Hough circle detection



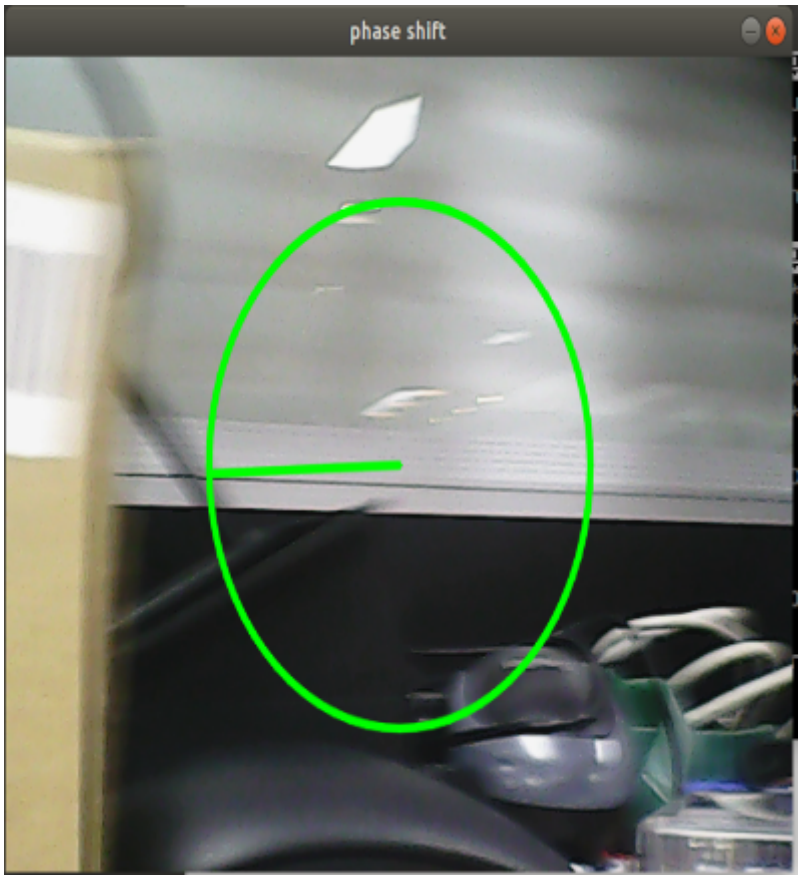
- Hough linear detection

The lower the threshold, the more lines there are and the easier it is for the picture to freeze.



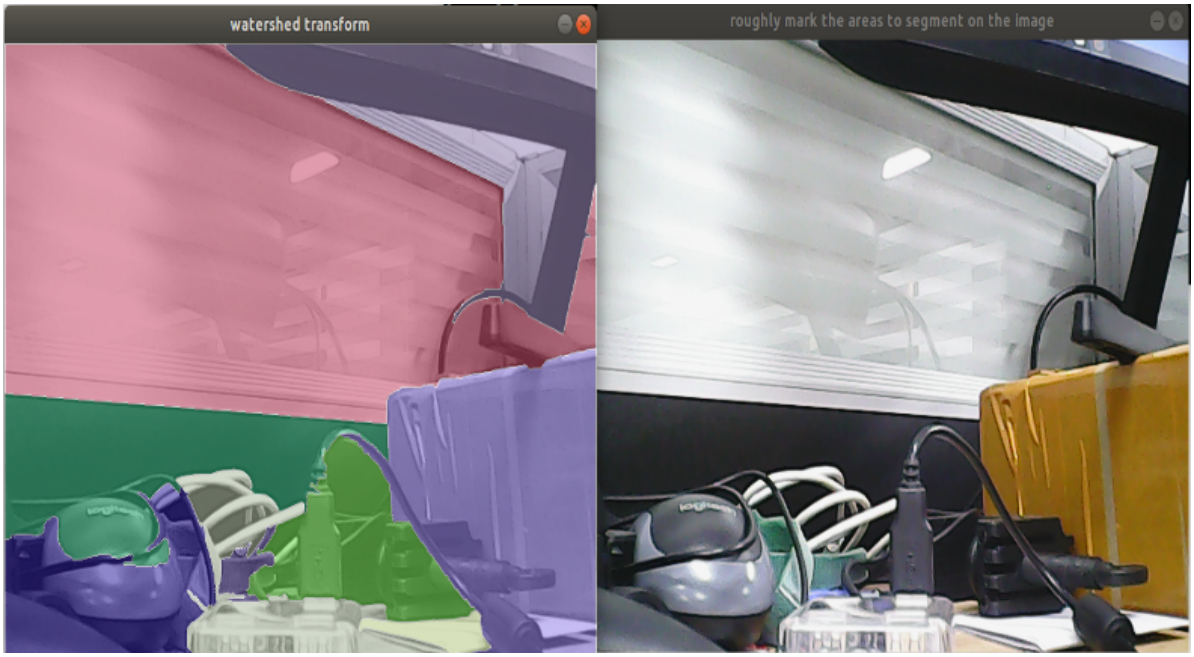
- Phase dependent displacement detection

The faster the camera moves, the larger the radius of the circle.



- Watershed segmentation algorithm

Use the mouse to select different objects and the system will automatically distinguish them.



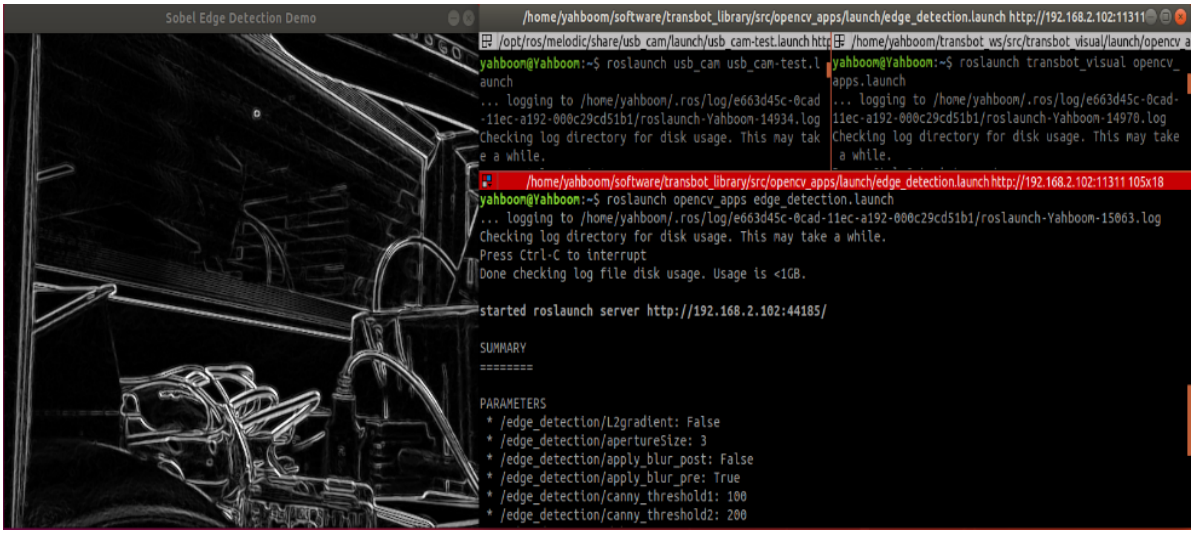
9.3. Node

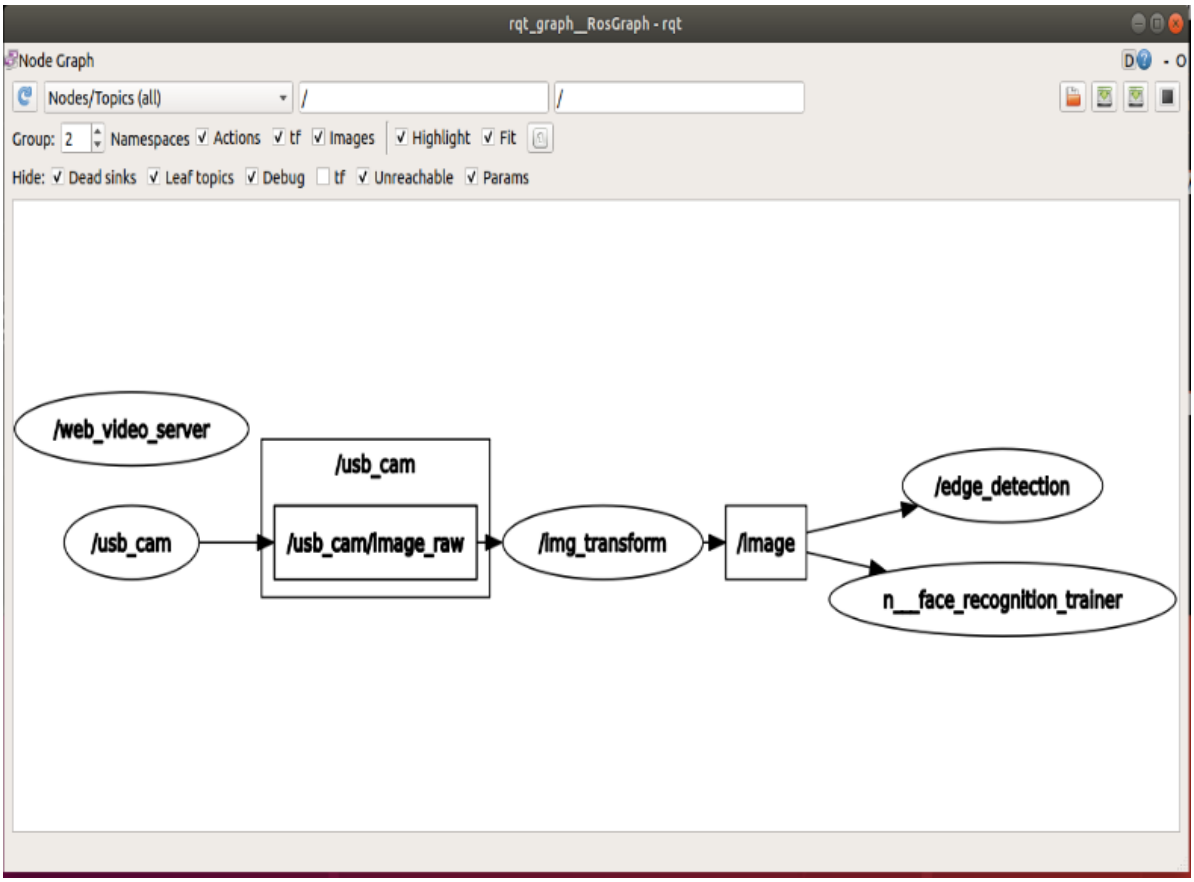
Each case in this section will have a topic for subscribing images and publishing images.

9.3.1. Edge detection algorithm

Parameters	Type	Default	Parse
~use_camera_info	bool	true	Subscribe to the topic [camera_info] and get the default coordinate system ID, otherwise use the image information directly.
~debug_view	bool	false	Whether to create a window to display the node image
~edge_type	int	0	Specify the edge detection method: 0: Sobel operator, 1: Laplacian operator, 2: Canny edge detection
~canny_threshold1	int	100	Specify the second canny threshold
~canny_threshold2	int	200	Specify the first canny threshold
~apertureSize	int	3	The aperture size of the Sobel operator.
~apply_blur_pre	bool	True	Whether to apply blur() to the input image
~postBlurSize	double	3.2	Input image aperture size

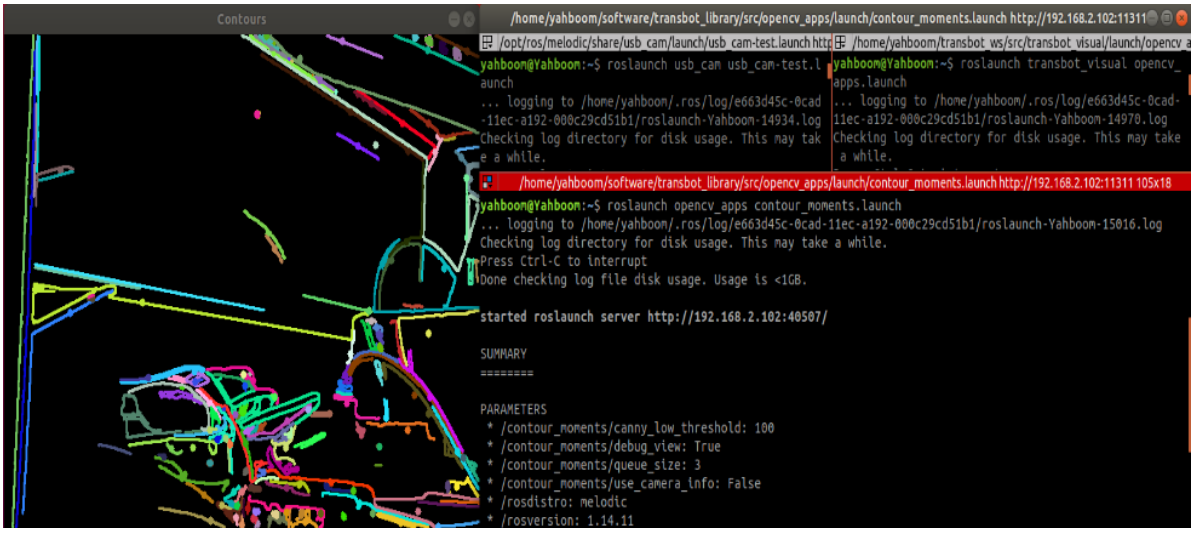
Parameters	Type	Default	Parse
~apply_blur_post	bool	False	Whether to apply GaussianBlur() to the input image
~L2gradient	bool	False	canny's parameters
~queue_size	int	3	Queue size





9.3.2. Contour moment

Parameters	Type	Default	Parse
~use_camera_info	bool	true	Subscribe to the topic [camera_info] and get the default coordinate system ID, otherwise use the image information directly.
~debug_view	bool	false	Whether to create a window to display the node image
~canny_low_threshold	int	0	Canny edge detection low threshold
~queue_size	int	3	Queue size



9.3.3. Face recognition

This case involves autonomous training and real-time recognition by collecting images of people in real time, and the steps are slightly complicated.

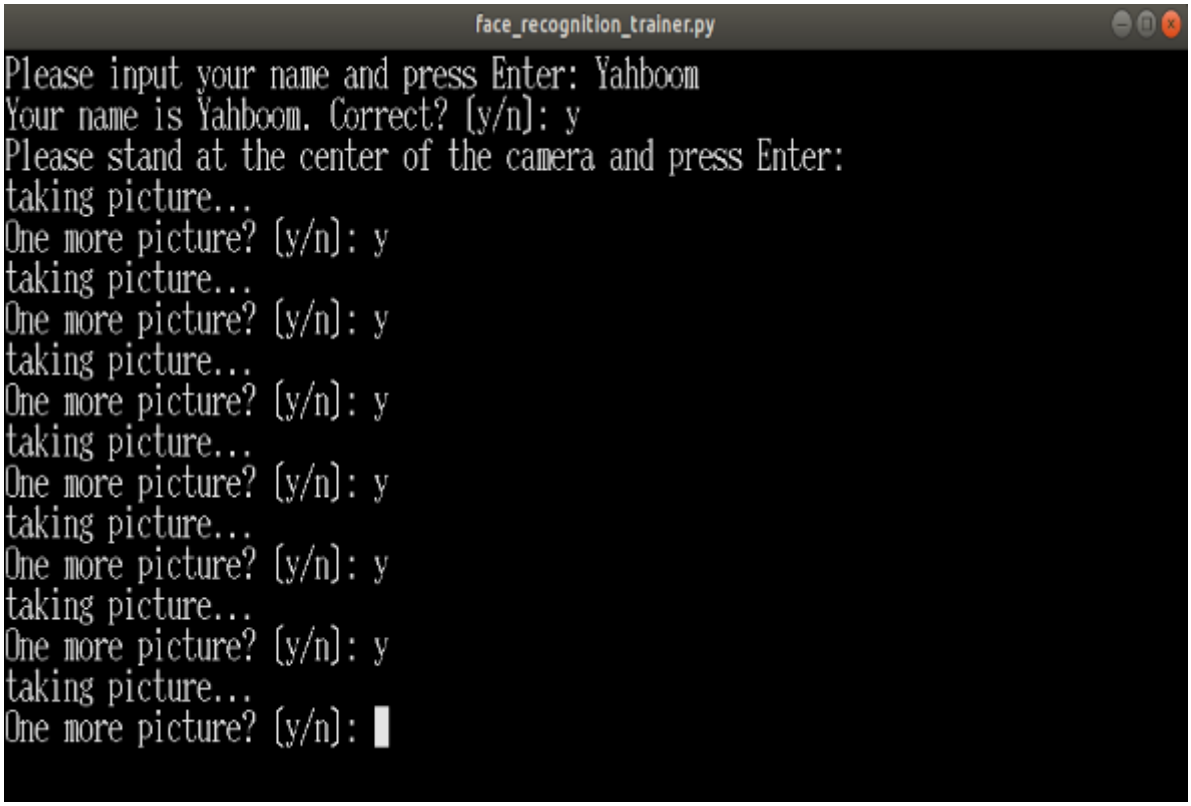
Parameters	Type	Default	Parse
~approximate_sync	bool	false	Subscribe to the topic [camera_info] and get the default coordinate system ID, otherwise use the image information directly.
~queue_size	int	100	The queue size of the subscription topic
~model_method	string	"eigen"	Face recognition method: "eigen", "fisher" or "LBPH"
~use_saved_data	bool	true	Load training data from ~data_dir path
~save_train_data	bool	true	Save the training data to the ~data_dir path for retraining
~data_dir	string	"~/opencv_apps/face_data"	Save training data path
~face_model_width	int	190	Width of training face image
~face_model_height	int	90	Height of training face image

Parameters	Type	Default	Parse
~face_padding	double	0.1	The padding ratio of each face
~model_num_components	int	0	Number of components for the face recognizer model (0 is considered unlimited)
~model_threshold	double	8000.0	Face recognition model threshold
~lbph_radius	int	1	Radius parameter (only for LBPH method)
~lbph_neighbors	int	8	Neighborhood parameters (only for LBPH method)
~lbph_grid_x	int	8	Grid x parameter (only for LBPH method)
~lbph_grid_y	int	8	Grid y parameter (only for LBPH method)
~queue_size	int	100	Image subscriber queue size

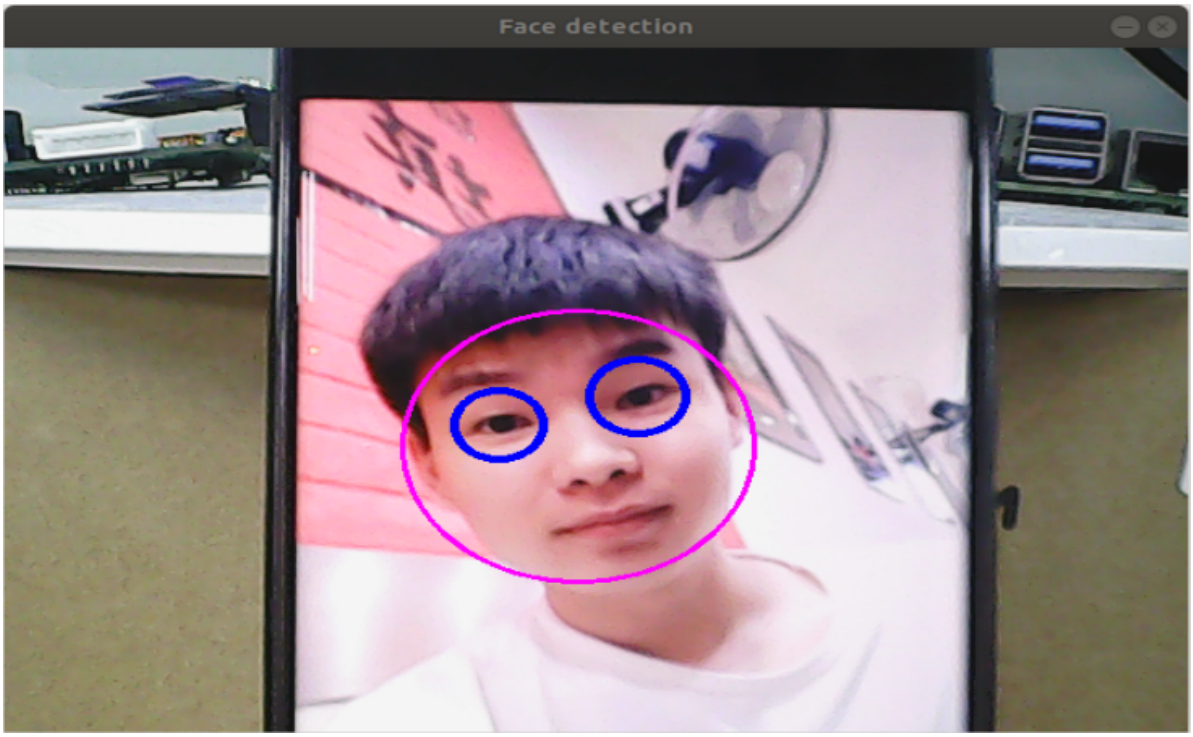
Steps:

1. First, enter the character's name after the colon in the picture below: Yahboom
2. Confirm name: y
3. Then place the face in the center of the image and click OK.
4. Add a photo in the loop: y, click to confirm.
5. To end image collection, enter: n and click Confirm.
6. Close the launch file and restart.

If you need to enter the recognized identifications, cycle through steps 1 to 5 until all identified persons have been entered, and then proceed to step six.



Step 3: Ensure that faces can be recognized



final recognition effect


/face_recognition/debug_image
Yahboom (1720.20)