

4、ROS+OpenCV application

4、ROS+OpenCV application

4.1、Overview

4.2、Usage

4.2.1、Start up

4.2.2、Display method

4.2.3、Effect show

4.3、Node

4.3.1、Edge detection algorithm

4.3.2、Contour moment

4.3.3、Face recognition

4.1、Overview

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

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

Function package: ~/orbbeec_ws/src/opencv_apps

Most of the code was originally from this website. <https://github.com/Itseez/opencv/tree/master/samples/cpp>

opencv_apps provides various nodes that run internally OpenCV's functionalities and publish the result as ROS topics. With opencv_apps, you can skip writing OpenCV application codes for a lot of its functionalities by simply running a launch file that corresponds to OpenCV's functionality you want.

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

Contents

1. Introduction, usage
2. Edge Detection Nodes
 1. edge_detection
 2. hough_lines
 3. hough_circles
3. Structural Analysis Nodes
 1. find_contours
 2. convex_hull
 3. general_contours
 4. contour_moments
4. People/Face Detection Nodes
 1. face_detection
 2. face_recognition
 3. people_detect
5. Motion Analysis Nodes
 1. goodfeature_track
 2. camshift
 3. fback_flow
 4. lk_flow
 5. phase_corr
 6. simple_flow
6. Object Segmentation Nodes
 1. segment_objects
 2. watershed_segmentation
7. Image Filter Nodes
 1. rgb_color_filter
 2. hls_color_filter
 3. hsv_color_filter
8. Simple Image Processing Nodes
 1. adding_images

4.2、 Usage

4.2.1、 Start up

Step 1: Start the camera

```
roslaunch astra_visual opencv_apps.launch
```

If the webpage cannot be viewed, check if there is **[web_video_server]** node, if not, run the following command

```
roslaunch web_video_server web_video_server
```

Step 2: Start the function of Opencv_apps

```
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 moment
```

<code>roslaunch opencv_apps convex_hull.launch</code>	<code># Polygon outline</code>
<code>roslaunch opencv_apps discrete_fourier_transform.launch</code>	<code># Discrete Fourier Transform Algorithm</code>
<code>roslaunch opencv_apps edge_detection.launch</code>	<code># Edge detection algorithm</code>
<code>roslaunch opencv_apps face_detection.launch</code>	<code># Face detection algorithm</code>
<code>roslaunch opencv_apps fback_flow.launch</code>	<code># Optical flow detection algorithm</code>
<code>roslaunch opencv_apps find_contours.launch</code>	<code># Contour detection</code>
<code>roslaunch opencv_apps general_contours.launch</code>	<code># General contour detection</code>
<code>roslaunch opencv_apps goodfeature_track.launch</code>	<code># Feature point tracking</code>
<code>roslaunch opencv_apps hls_color_filter.launch</code>	<code># HLS color filter</code>
<code>roslaunch opencv_apps hough_circles.launch</code>	<code># Hough circle detection</code>
<code>roslaunch opencv_apps hough_lines.launch</code>	<code># Hough line detection</code>
<code>roslaunch opencv_apps hsv_color_filter.launch</code>	<code># HSV color filter</code>
<code>roslaunch opencv_apps lk_flow.launch</code>	<code># LK optical flow algorithm</code>
<code>roslaunch opencv_apps people_detect.launch</code>	<code># Human detection algorithm</code>
<code>roslaunch opencv_apps phase_corr.launch</code>	<code># Phase correlation displacement detection</code>
<code>roslaunch opencv_apps pyramids.launch</code>	<code># Image pyramid sampling algorithm</code>
<code>roslaunch opencv_apps rgb_color_filter.launch</code>	<code># RGB color filtering</code>
<code>roslaunch opencv_apps segment_objects.launch</code>	<code># Clear background detection algorithm</code>
<code>roslaunch opencv_apps simple_flow.launch</code>	<code># Simplified optical flow algorithm</code>
<code>roslaunch opencv_apps smoothing.launch</code>	<code># Simple filter</code>
<code>roslaunch opencv_apps threshold.launch</code>	<code># Threshold image processing</code>
<code>roslaunch opencv_apps watershed_segmentation.launch</code>	<code># watershed segmentation algorithm</code>

Almost every function case will have a parameter [debug_view], boolean type, whether to use Opencv to display the picture, it is displayed by default.

If you don't need to display it, set it to [False], for example

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

4.2.2、Display method

- `rqt_image_view`

Enter the following command to select the corresponding topic

```
rqt_image_view
```

- `opencv`

The system displays it by default.

- Web view

(At the same LAN) Enter IP+port in the browser, for example:

192.168.2.102:8080

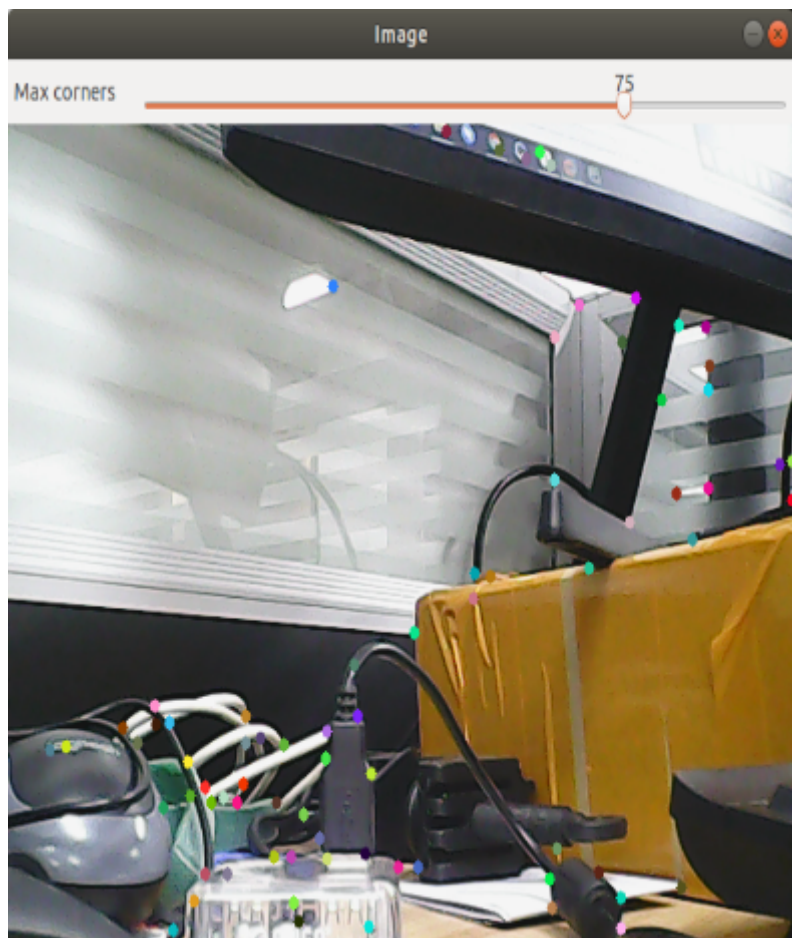
4.2.3、Effect show

- Optical flow detection algorithm

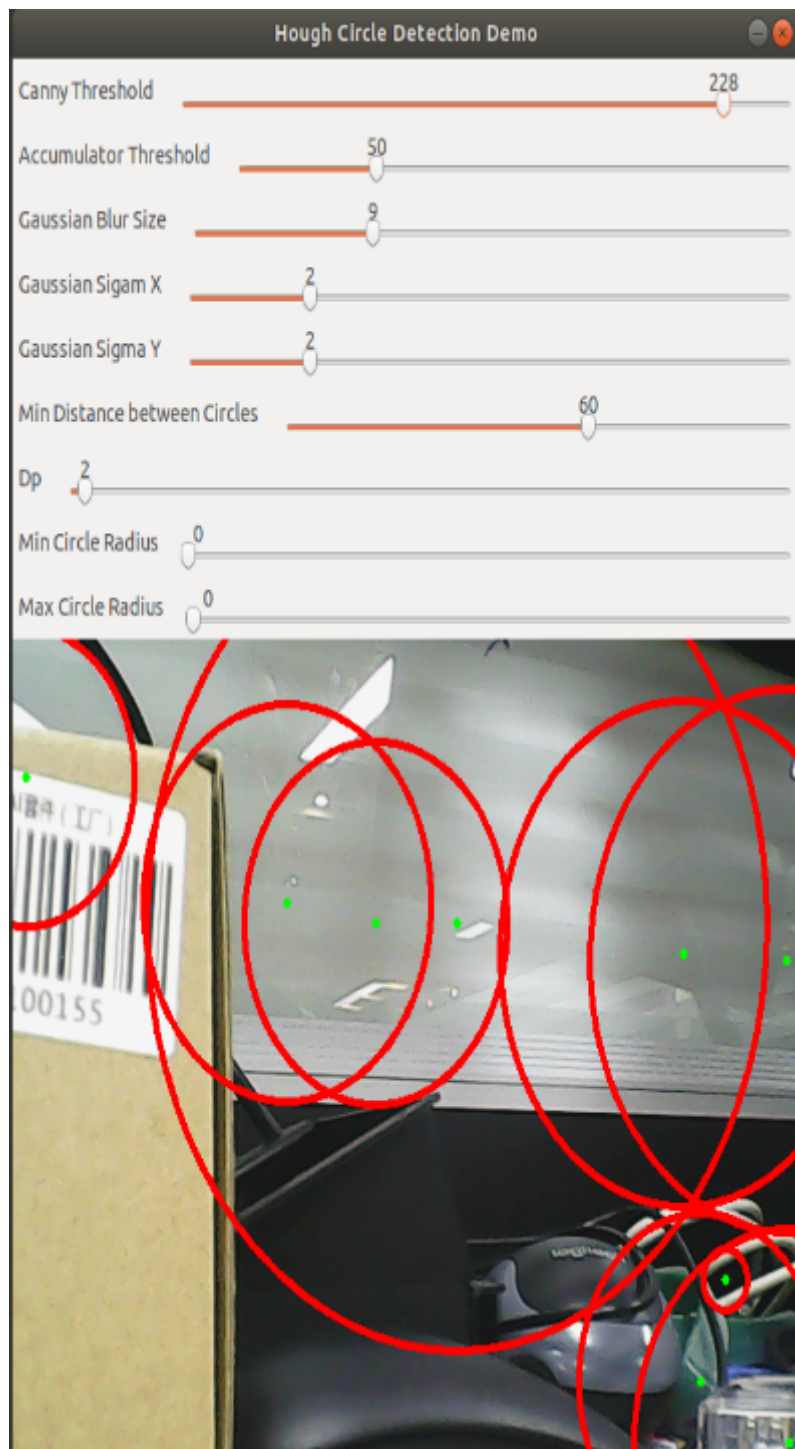
Move the screen and observe the phenomenon.



- Feature point tracking

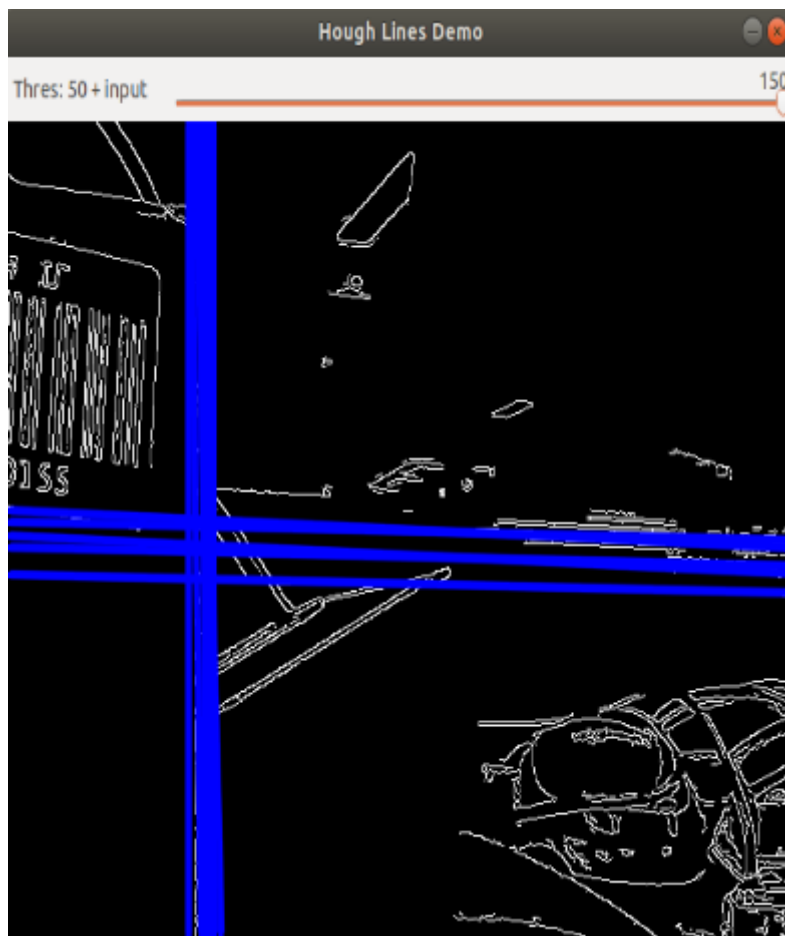


- Hough circle detection



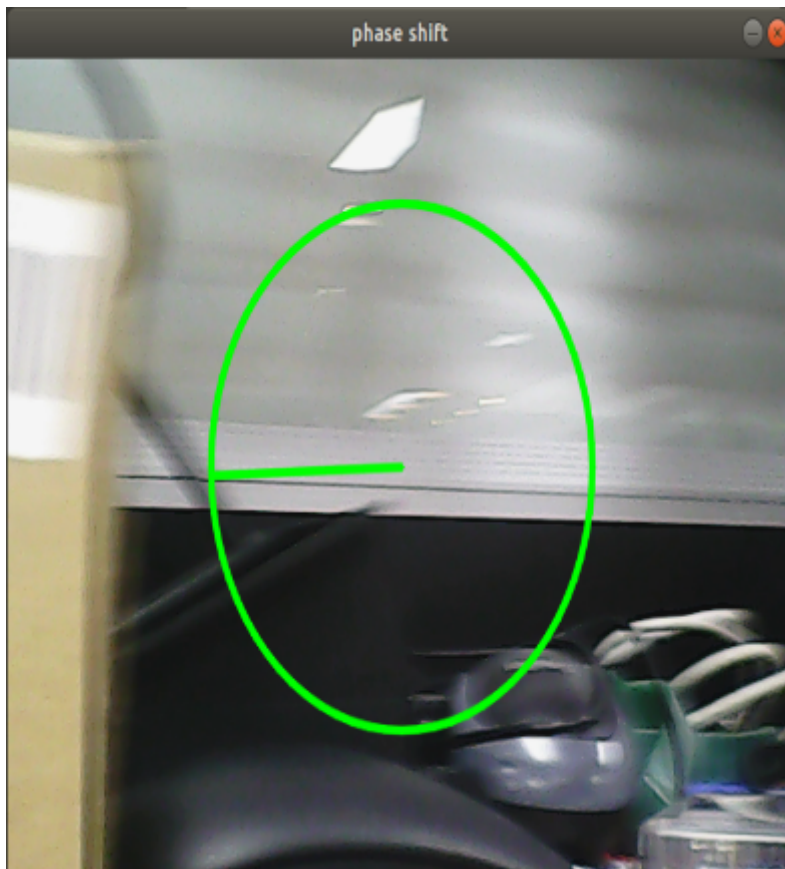
- Hough line detection

The lower the threshold, the more lines there are, and the more easily the picture gets stuck.



- Phase correlation displacement detection

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



- Watershed segmentation algorithm

Use the mouse to select different objects, the system automatically distinguishes them.

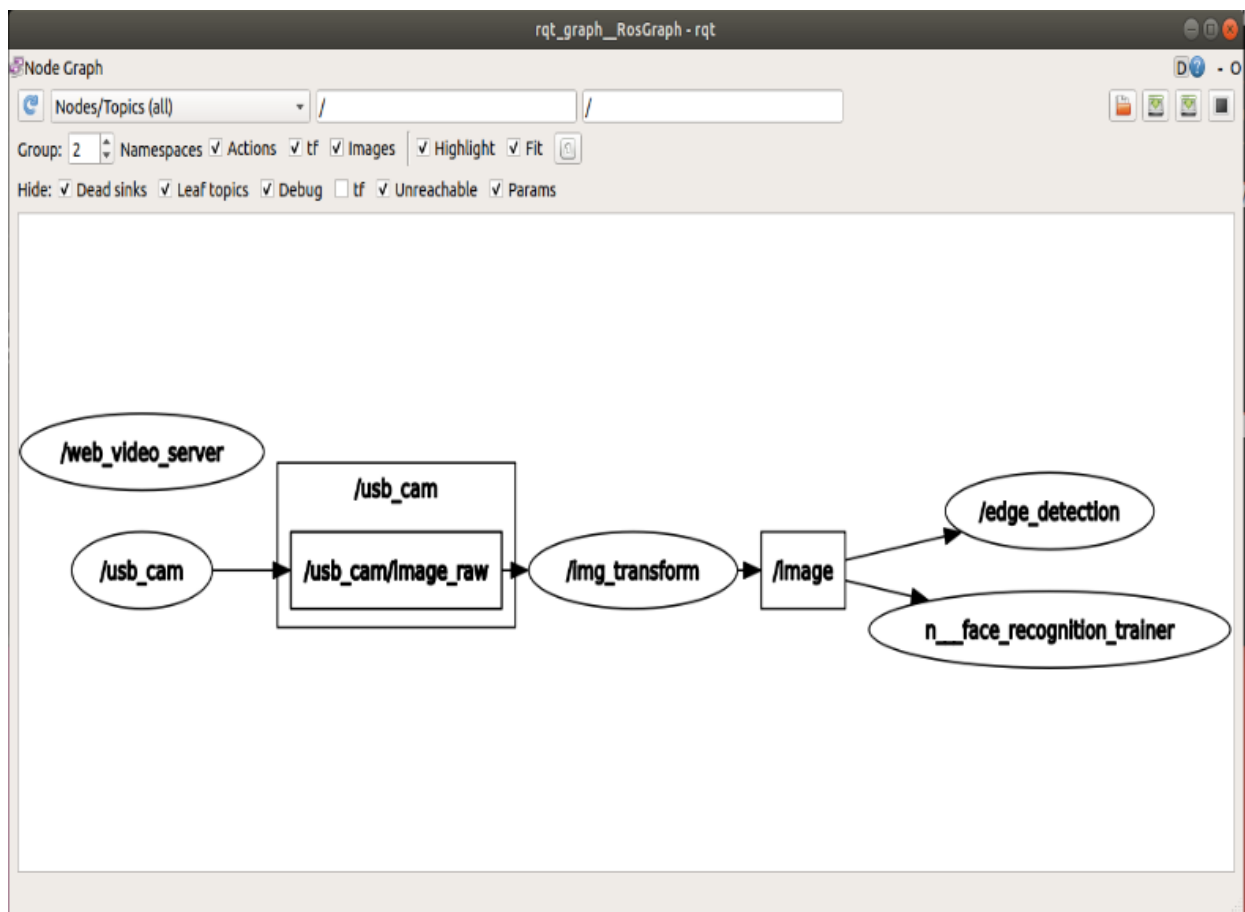
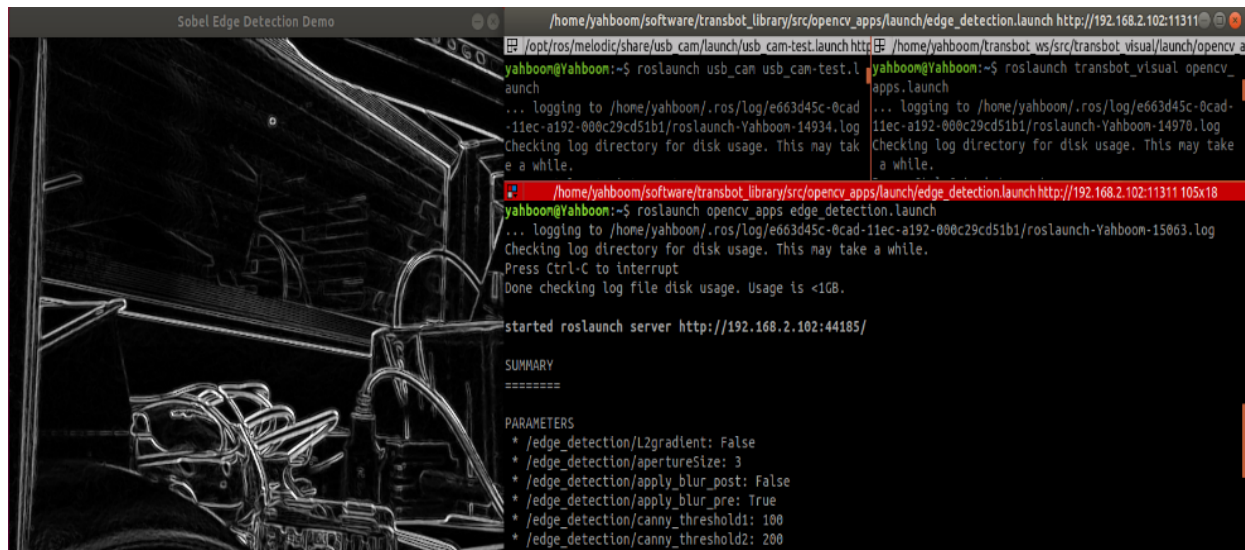


4.3、 Node

4.3.1、 Edge detection algorithm

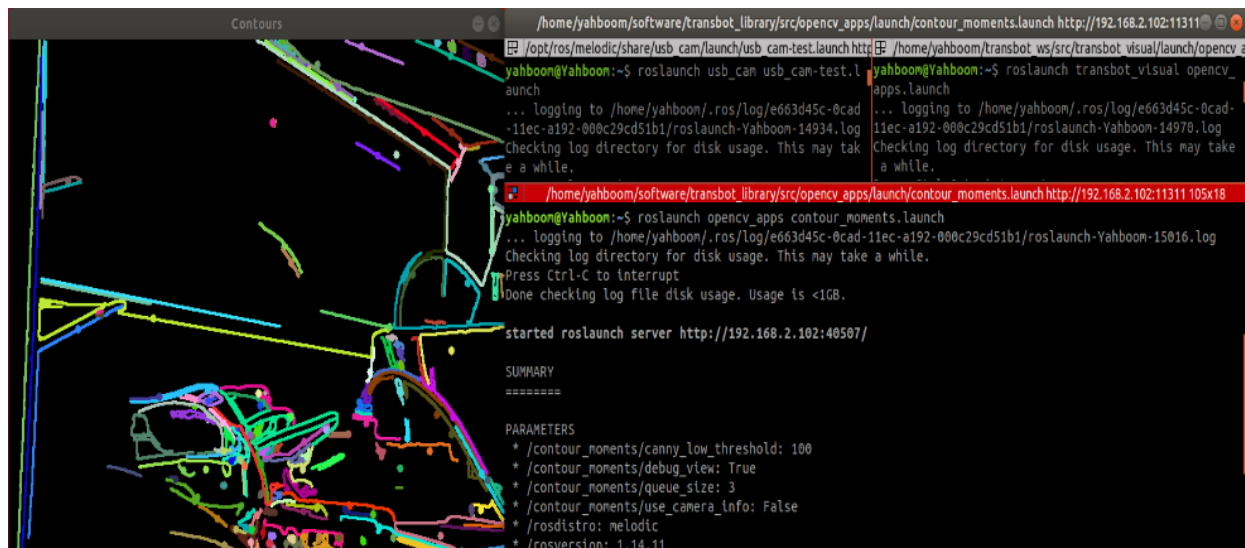
Parameter	Type	Default	Analyze
~use_camera_info	bool	true	Subscribe to the topic [camera_info] to get the default coordinate system ID, otherwise use the image information directly.
~debug_view	bool	false	Whether to create a window to display the node image
~edge_type	int	0	Specify the edge detection 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
~apply_blur_post	bool	False	Whether to apply GaussianBlur() to the input image

Parameter	Type	Default	Analyze
~L2gradient	bool	False	Parameters of canny
~queue_size	int	3	Queue size



4.3.2、Contour moment

Parameter	Type	Default	Analyze
~use_camera_info	bool	true	Subscribe to the topic [camera_info] to 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



4.3.3、Face recognition

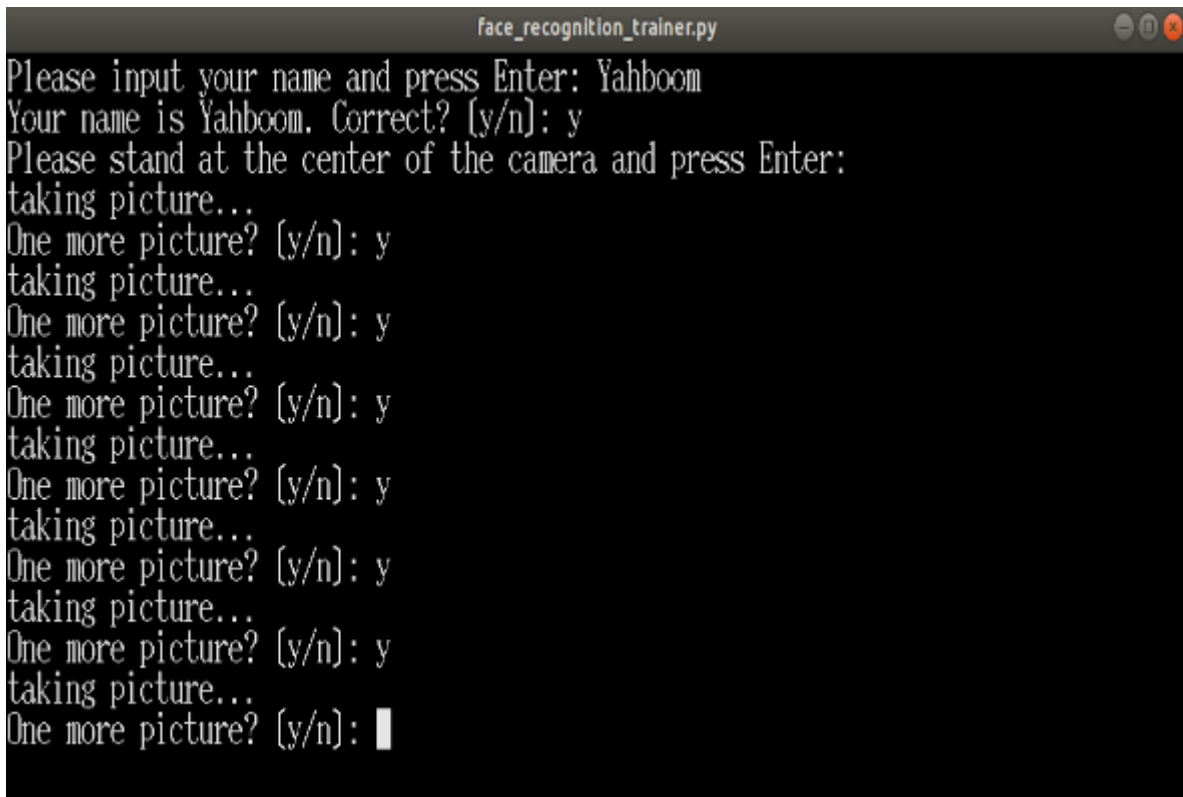
Parameter	Type	Default	Analyze
~approximate_sync	bool	false	Subscribe to the topic [camera_info] to get the default coordinate system ID, otherwise use the image information directly.
~queue_size	int	100	Queue size for subscribed topics
~model_method	string	"eigen"	Face recognition method: "eigen", "fisher" or "LBPH"

Parameter	Type	Default	Analyze
~use_saved_data	bool	true	Load training data from the ~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 the training data path
~face_model_width	int	190	Train the width of the face image
~face_model_height	int	90	Training the height of the face image
~face_padding	double	0.1	Fill ratio of each face
~model_num_components	int	0	The number of components of the face recognizer model (0 is considered unlimited)
~model_threshold	double	8000.0	Face recognition model threshold
~lbph_radius	int	1	Radius parameter (only applicable to LBPH method)
~lbph_neighbors	int	8	Neighborhood parameters (only applicable to LBPH method)
~lbph_grid_x	int	8	Grid x parameters (only applicable to LBPH method)
~lbph_grid_y	int	8	Grid y parameter (only applicable to LBPH method)
~queue_size	int	100	Image subscriber queue size

Steps:

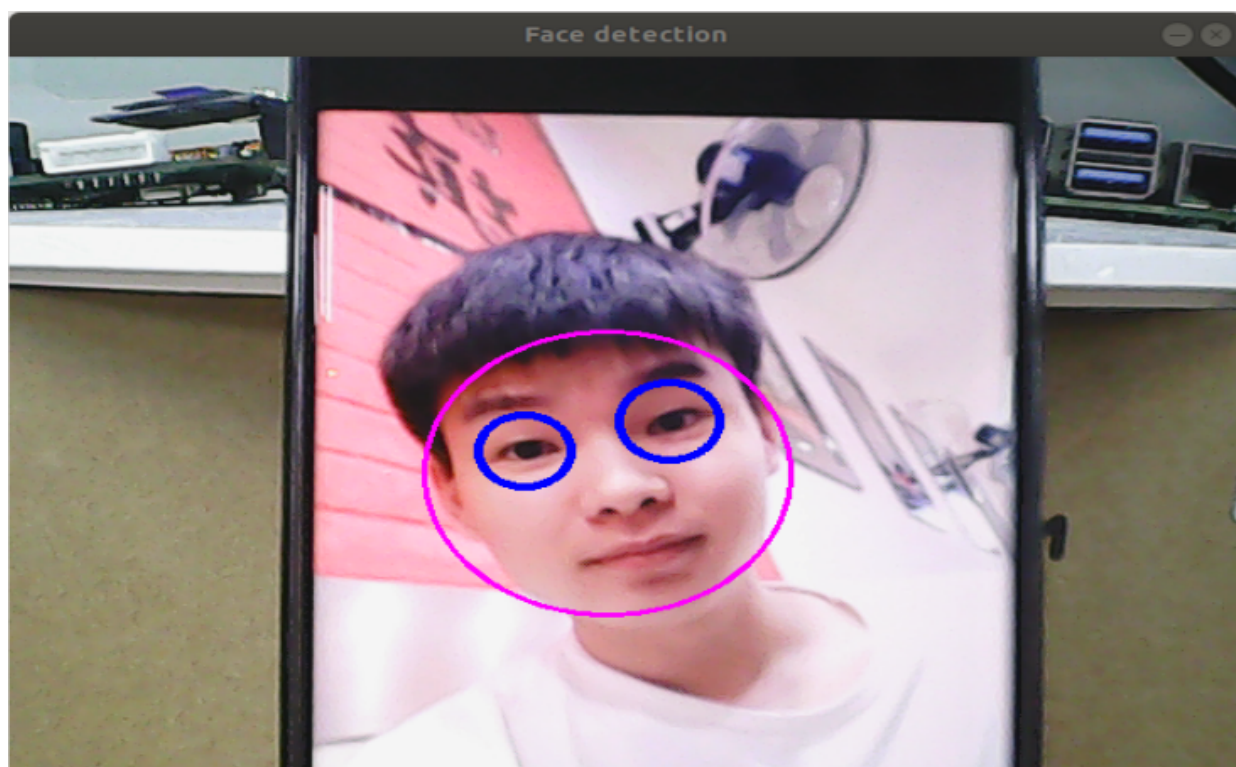
1. First, after the colon in the figure below, enter the character's name: Yahboom
2. Confirm name: y
3. Then place the face in the center of the image and click OK.
4. Cycle to add a photo: y, click to confirm.
5. To end the picture collection, enter: n and click to confirm.
6. Close the launch file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

If you need to enter the recognition, cycle 1~5 in turn until all the recognition personnel are entered, and then perform step 6.



```
face_recognition_trainer.py
Please input your name and press Enter: Yahboom
Your name is Yahboom. Correct? [y/n]: y
Please stand at the center of the camera and press Enter:
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]:
```

Step 3: Ensure that the face can be recognized



Recognition effect

