2. Opencv application--face recognition

2.1 Overview

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

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

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

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

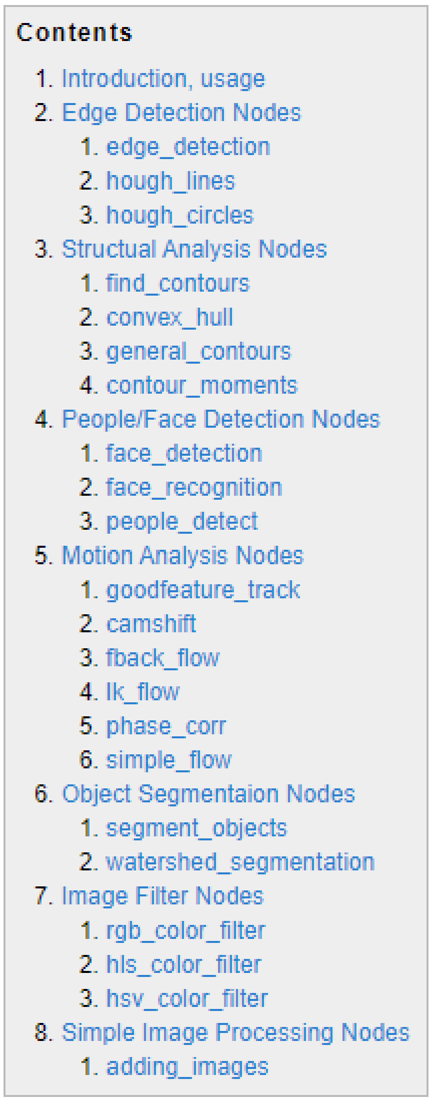
The topic subscribed by this function package is [/image]. What we need to do is to open the camera node and write a node to convert the camera topic into [/image] and publish the [/image] topic.

The path of the node that opens the camera and publishes the [/image] topic:

```
~/ascam_ws/src/ascam_visual/scripts/pub_image.py
```

The opencv_apps program provides various nodes that run opencv functions internally and publish the results to ROS topics. 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 code for these functions.

ROS Wiki has relevant node analysis, topic subscription and topic publishing of the corresponding node, and related parameter introduction. For details, please see ROS WiKi.



Contents

1. Introduction, usage
2. Edge Detection Nodes
   1. edge_detection
   2. hough_lines
   3. hough_circles
3. Structual 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 Segmentaion 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

2.2, Use

Step 1: Start the camera

```
roslaunch ascam_visual opencv_apps.launch
```

- img_flip parameter: whether the image needs to be flipped horizontally, the default is false.

The [usb_cam-test.launch] file opens the [web_video_server] node by default, and you can directly use the [IP:8080] web page to view images in real time.

Step 2: Start the face recognition function

```
roslaunch opencv_apps face_recognition.launch # Face recognition
```

Almost every case of ros+opencv application has a parameter [debug_view], Boolean type, whether to use Opencv to display the image, which is displayed by default.

If it does not need to be displayed, set it to [False], for example

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

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

2.3, Display method

- rqt_image_view

Enter the following command and select the corresponding topic

```
rqt_image_view
```

- opencv

The system displays by default, no processing is required.

- Web viewing

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

```
192.168.2.116:8080
```

For specific IP, use your current virtual machine IP.

2.4, Face recognition display

This case is to collect people's images for self-training and real-time recognition, and the steps are slightly complicated.

Parameter	Type	Default	Analysis
~approximate_sync	bool	false	Subscribe to the topic [camera_info] to obtain the default coordinate system ID, otherwise use the image information directly.
~queue_size	int	100	Queue size for subscribing to topics
~model_method	string	"eigen"	Face recognition method: "eigen", "fisher" or "LBPH"
~use_saved_data	bool	true	Load training data from ~data_dir

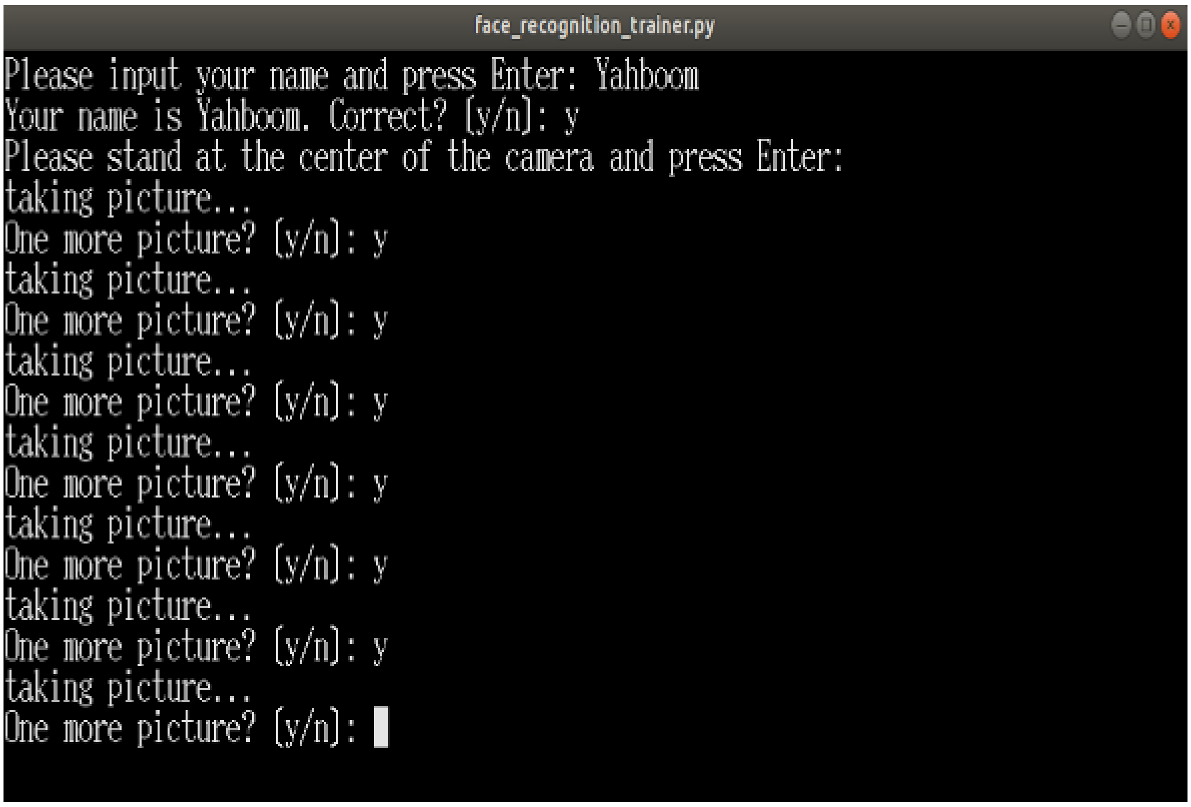
Parameter	Type	Default	Analysis
~save_train_data	bool	true	Save training data to ~data_dir for retraining
~data_dir	string	"~/opencv_apps/face_data"	Path to save training data
~face_model_width	int	190	Width of training face images
~face_model_height	int	90	Height of training face images
~face_padding	double	0.1	Padding ratio for 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 for LBPH method)
~lbph_neighbors	int	8	Neighborhood parameter (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 name of the person after the colon in the figure below: Yahboom
2. Confirm the name: y
3. Then put the face in the center of the image and click Confirm.
4. Loop to add a photo: y, click Confirm.
5. End the image collection, enter: n, and click OK.

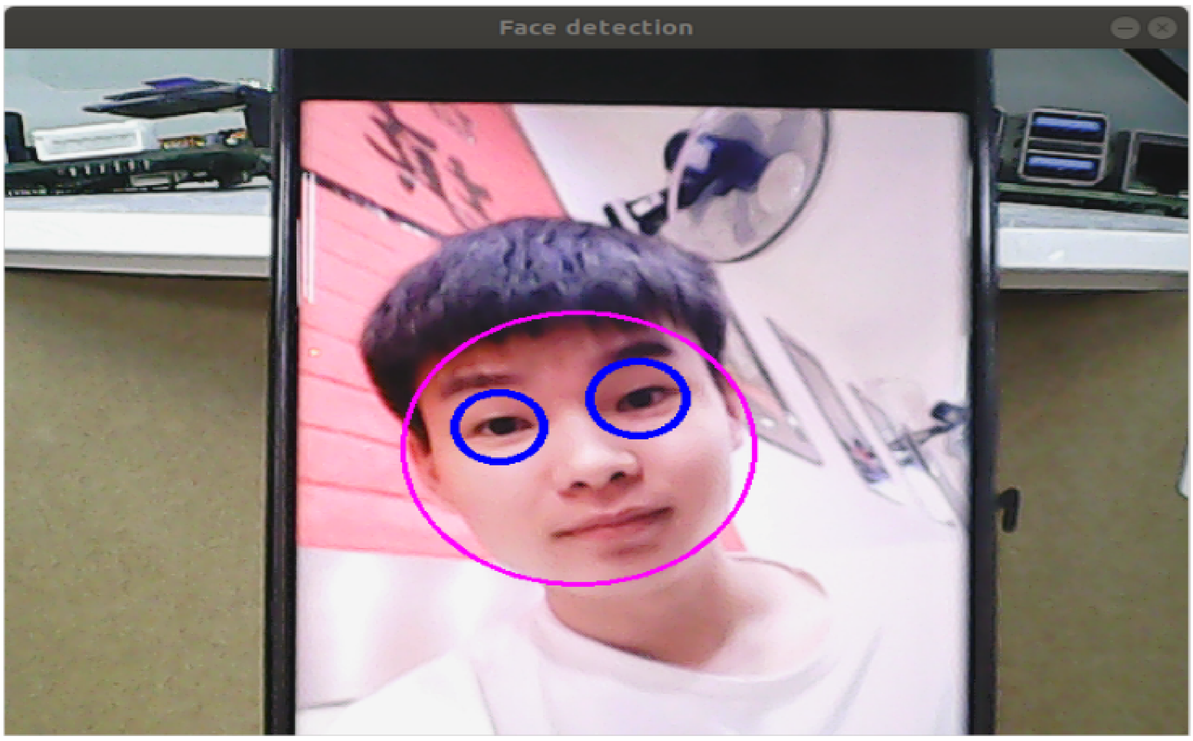
6. Close the launch file and restart.

If you need to enter the recognition, loop 1 to 5 in sequence until all the recognized people are entered, and then execute step 6.

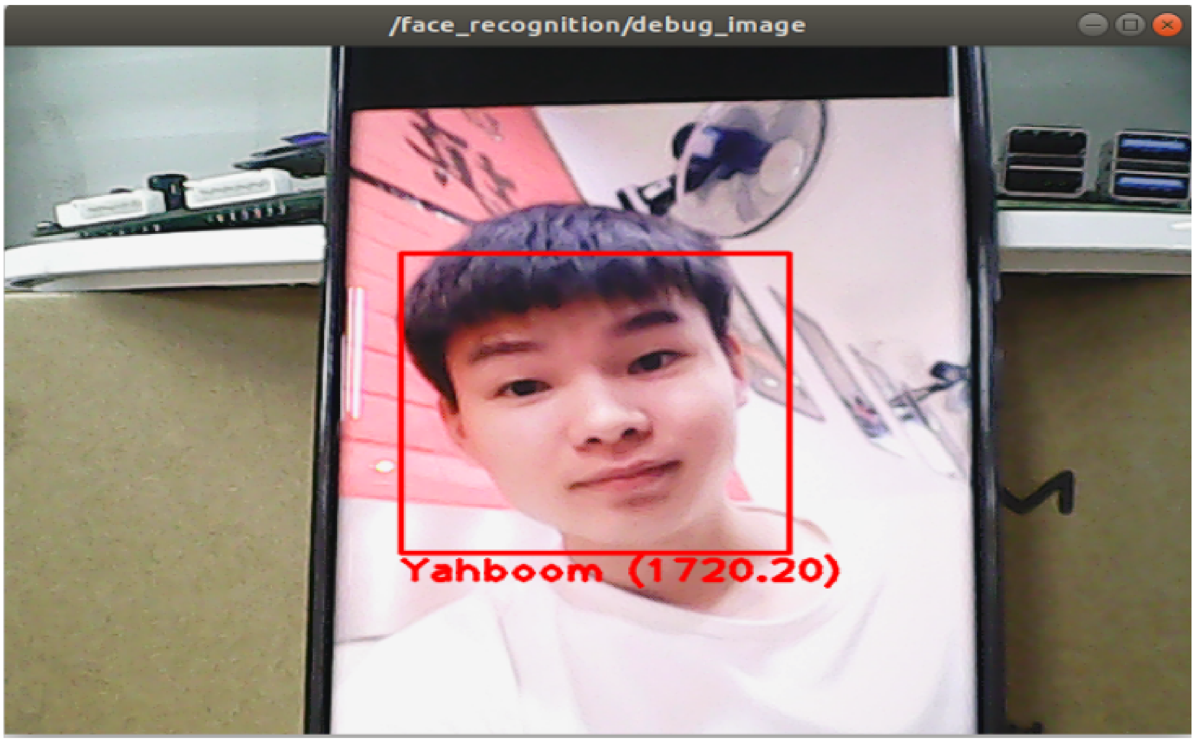


```
Please input your name and press Enter: Yahboom
Your name is Yahboom. Correct? [y/n]: y
Please stand at the center of the camera and press Enter:
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...
One more picture? [y/n]: y
taking picture...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



Final recognition effect


/face_recognition/debug_image
Yahboom (1720.20)