本次作业第一题，第二题安装 tensorflow，但是一直报错，思路应该类似
在第一个终端运行
roslaunch robot_vision usb_cam.launch
第二个终端运行
roslaunch robot_vision face_detector.launch
第三个终端运行
rqt_image_view
第四个终端运行
roslaunch mbot_gazebo view_mbot_with_kinect_gazebo.launch

关键代码（两个）
本质就是发布脸部变化的坐标话题
然后根据上述话题在 test1.cpp 里面进行处理，对机器人模型速度指令进行发布

face_detector.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import rospy
import cv2
import numpy as np
from sensor_msgs.msg import Image, RegionOfInterest
from cv_bridge import CvBridge, CvBridgeError
from test1.msg import Size

class faceDetector:
  def __init__(self):
    rospy.on_shutdown(self.cleanup);

    # 创建 cv_bridge
    self.bridge = CvBridge()
    self.size = Size()
    self.image_pub = rospy.Publisher("cv_bridge_image", Image, queue_size=1)
    self.size_pub = rospy.Publisher("Size", Size, queue_size=1)

    # 获取 haar 特征的级联表的 XML 文件，文件路径在 launch 文件中传入
    cascade_1 = rospy.get_param("~cascade_1", "")
    cascade_2 = rospy.get_param("~cascade_2", "")

    # 使用级联表初始化 haar 特征检测器
    self.cascade_1 = cv2.CascadeClassifier(cascade_1)
    self.cascade_2 = cv2.CascadeClassifier(cascade_2)

    # 设置级联表的参数，优化人脸识别，可以在 launch 文件中重新配置
    self.haar_scaleFactor  = rospy.get_param("~haar_scaleFactor", 1.2)
    self.haar_minNeighbors = rospy.get_param("~haar_minNeighbors", 2)
    self.haar_minSize      = rospy.get_param("~haar_minSize", 40)
    self.haar_maxSize      = rospy.get_param("~haar_maxSize", 60)
    self.color = (50, 255, 50)
```

```
    # 初始化订阅 rgb 格式图像数据的订阅者，此处图像 topic 的话题名可以在 launch 文件中重映射
    self.image_sub = rospy.Subscriber("input_rgb_image", Image, self.image_callback,
queue_size=1)

  def image_callback(self, data):
    # 使用 cv_bridge 将 ROS 的图像数据转换成 OpenCV 的图像格式
    try:
      cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
      frame = np.array(cv_image, dtype=np.uint8)
    except CvBridgeError, e:
      print e

    # 创建灰度图像
    grey_image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 创建平衡直方图，减少光线影响
    grey_image = cv2.equalizeHist(grey_image)

    # 尝试检测人脸
    faces_result = self.detect_face(grey_image)

    # 在 opencv 的窗口中框出所有人脸区域
    if len(faces_result)>0:
      for face in faces_result:
        x, y, w, h = face
        self.size.width = abs(int(w))
        self.size.high = abs(int(h))
        self.size.x = abs(int(x))
        self.size_pub.publish(self.size)
        cv2.rectangle(cv_image, (x, y), (x+w, y+h), self.color, 2)

    # 将识别后的图像转换成 ROS 消息并发布
    self.image_pub.publish(self.bridge.cv2_to_imgmsg(cv_image, "bgr8"))

  def detect_face(self, input_image):
    # 首先匹配正面人脸的模型
    if self.cascade_1:
      faces = self.cascade_1.detectMultiScale(input_image,
          self.haar_scaleFactor,
          self.haar_minNeighbors,
          cv2.CASCADE_SCALE_IMAGE,
          (self.haar_minSize, self.haar_maxSize))

    # 如果正面人脸匹配失败，那么就尝试匹配侧面人脸的模型
    if len(faces) == 0 and self.cascade_2:
      faces = self.cascade_2.detectMultiScale(input_image,
          self.haar_scaleFactor,
          self.haar_minNeighbors,
```

```
            cv2.CASCADE_SCALE_IMAGE,
            (self.haar_minSize, self.haar_maxSize))

        return faces

    def cleanup(self):
        print "Shutting down vision node."
        cv2.destroyAllWindows()

if __name__ == '__main__':
    try:
        # 初始化 ros 节点
        rospy.init_node("face_detector")
        faceDetector()
        rospy.loginfo("Face detector is started..")
        rospy.loginfo("Please subscribe the ROS image.")
        rospy.spin()
    except KeyboardInterrupt:
        print "Shutting down face detector node."
        cv2.destroyAllWindows()
```

test1.cpp

```
#include "ros/ros.h"
#include "test1/Size.h"
#include "geometry_msgs/Twist.h"
#include "std_msgs/Int32.h"

int flag = 0;
std_msgs::Int32 size_width, size_high, size_x;

class SubscribeAndPublish
{
public:
  SubscribeAndPublish()
  {
    pub = n.advertise<geometry_msgs::Twist>("cmd_vel", 1);
    sub = n.subscribe("Size", 1, &SubscribeAndPublish::callback, this);
  }

  void callback(const test1::Size::ConstPtr& size)
  {
     if(flag==0){
        size_width.data = size->width;
        size_high.data = size->high;
        size_x.data = size->x;
        flag = 1;
     }
     else{
```

```
      if((size->width>size_width.data+50)&&(size->high>size_high.data+50)){
       ROS_INFO("go forward");
       //ROS_INFO("size->width:%d size_width:%d", size->width, size_width.data);
       size_width.data = size->width;
       size_high.data = size->high;
       size_x.data = size->x;
       geometry_msgs::Twist msg;
       msg.linear.x = 0.3;
       msg.angular.z = 0;
       pub.publish(msg);
      }
      else if((size->width<size_width.data-50)&&(size->high<size_high.data-50)){
       ROS_INFO("go back");
       size_width.data = size->width;
       size_high.data = size->high;
       size_x.data = size->x;
       geometry_msgs::Twist msg;
       msg.linear.x = -0.3;
       msg.angular.z = 0;
       pub.publish(msg);
      }
      else if( (size->x + size->width/2) < (size_x.data + size_width.data/2 - 50) ){
       ROS_INFO("turn right");
       size_width.data = size->width;
       size_high.data = size->high;
       size_x.data = size->x;
       geometry_msgs::Twist msg;
       msg.linear.x = 0;
       msg.angular.z = -0.3;
       pub.publish(msg);
      }
      else if( (size->x + size->width/2) > (size_x.data + size_width.data/2 + 50) ){
       ROS_INFO("turn left");
       size_width.data = size->width;
       size_high.data = size->high;
       size_x.data = size->x;
       geometry_msgs::Twist msg;
       msg.linear.x = 0;
       msg.angular.z = 0.3;
       pub.publish(msg);
      }
    }
  }

private:
  ros::NodeHandle n;
  ros::Publisher pub;
  ros::Subscriber sub;
```

```
};

int main(int argc, char **argv)
{
 ros::init(argc, argv, "test1");
 SubscribeAndPublish SAPObject;
 ros::spin();

 return 0;
}
```