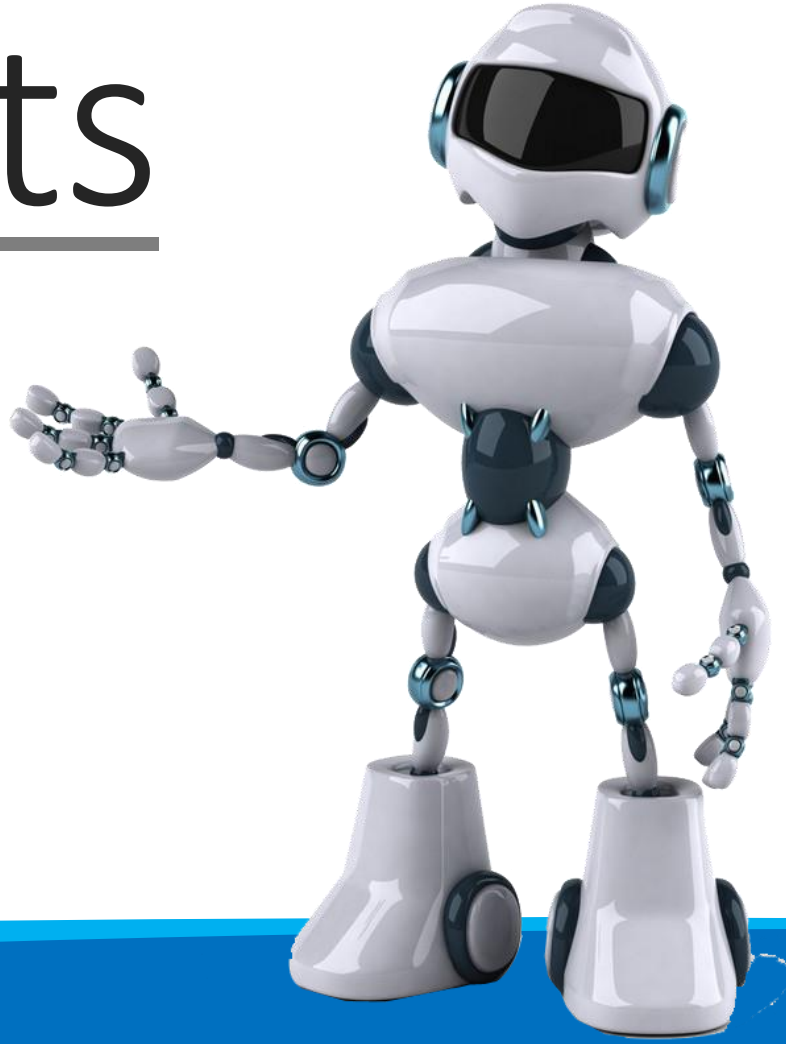
# Intelligent Robots

## LAB7

郝祁 Hao Qi
Email:haoq@sustech.edu.cn

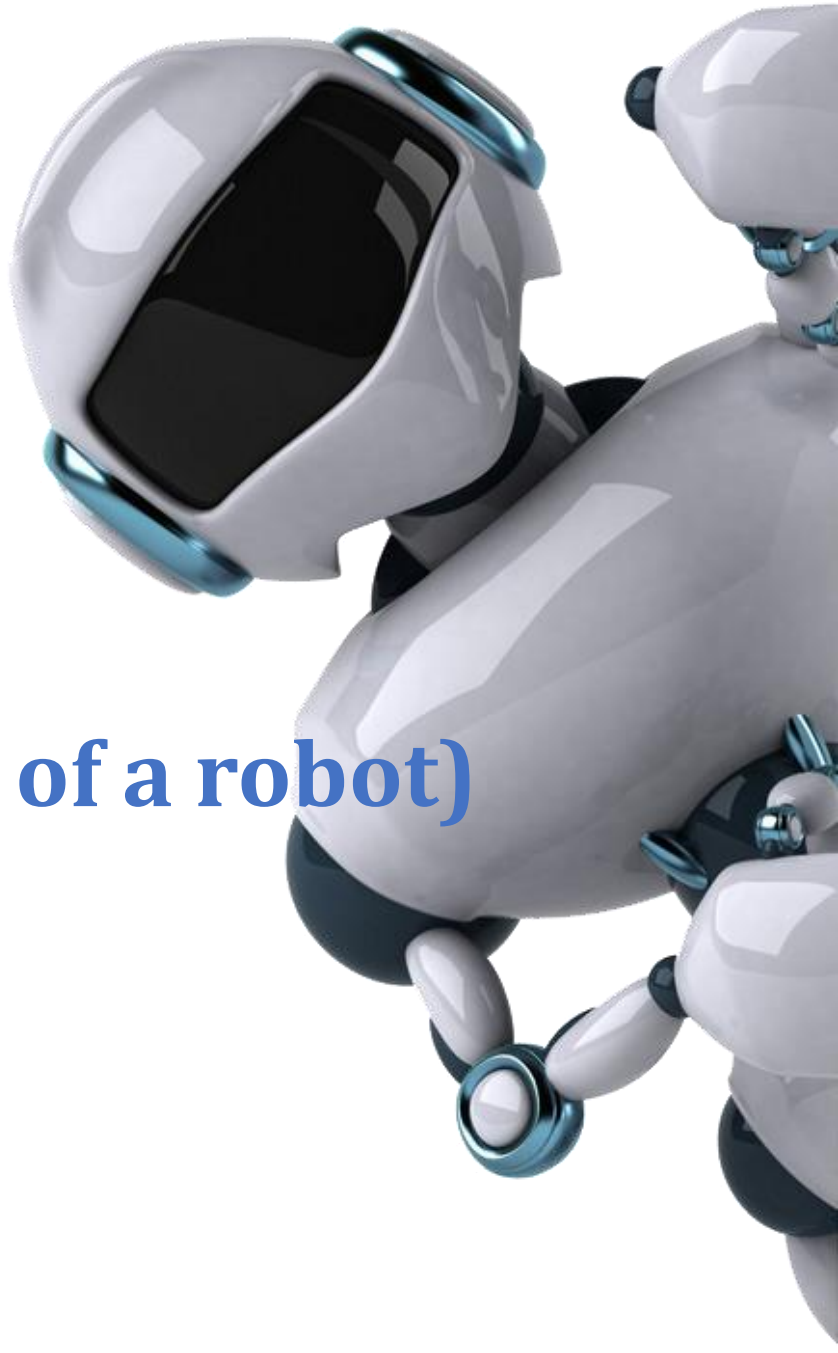


# OBJECTIVES

01 Obstacle avoidance

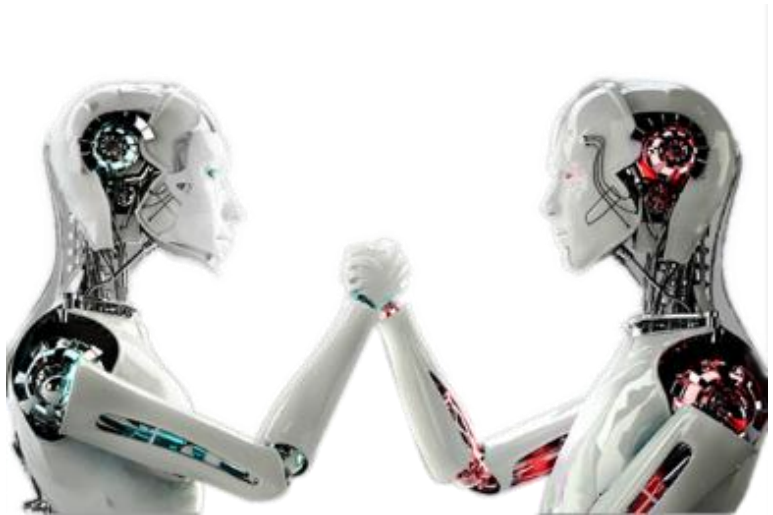
02 ROS+Opencv(Computer vision of a robot)

03 YOLO ROS



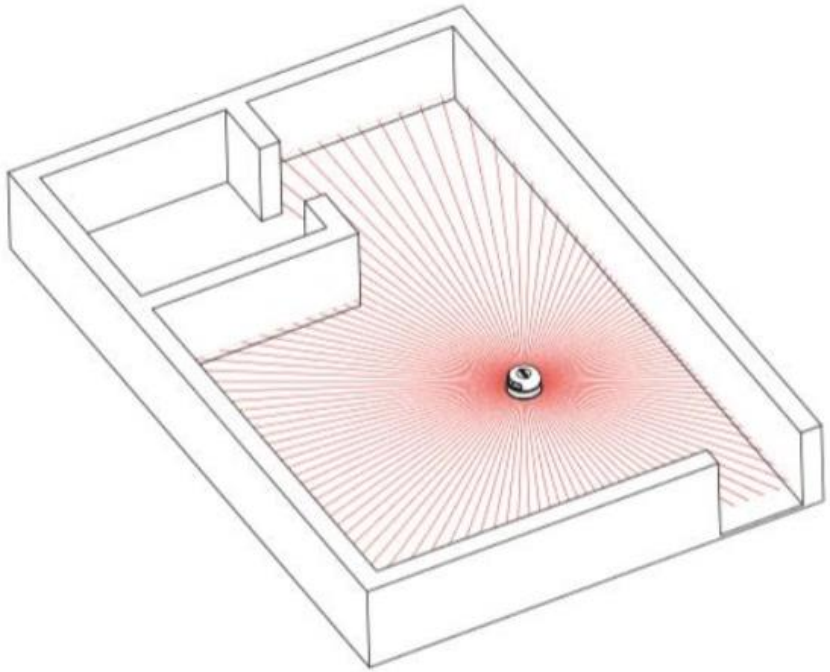
PART ONE

# Obstacle avoidance for ro

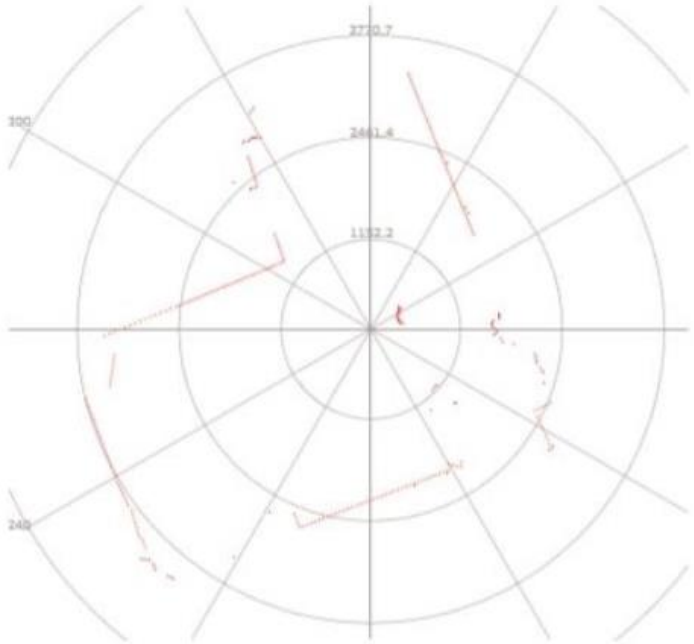


# Obstacle avoidance for robots

避障是指机器人在行走过程中，通过传感器感知到其路线规划上存在的动态或静态障碍物，按照一定的算法实时更新路径，避开障碍物，最终到达目的地。



扫描所在环境



RPLIDAR产生的环境轮廓数据

# Obstacle avoidance for robots

```
jyh@jyh:~/Downloads$ rostopic list
/camera/camera_info
/camera/image
/camera/image/compressed
/camera/image/compressed/parameter_descriptions
/camera/image/compressed/parameter_updates
/camera/image/compressedDepth
/camera/image/compressedDepth/parameter_descriptions
/camera/image/compressedDepth/parameter_updates
/camera/image/theora
/camera/image/theora/parameter_descriptions
/camera/image/theora/parameter_updates
/camera/parameter_descriptions
/camera/parameter_updates
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/joint_states
/odom
/rosout
/rosout_agg
/scan
/tf
/tf_static
/usb_cam/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descriptions
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
jyh@jyh:~/Downloads$
```

```
jyh@jyh:~/Downloads$ rostopic type /scan
sensor_msgs/LaserScan
jyh@jyh:~/Downloads$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

```
# 测量的激光扫描角度，逆时针为正
# 设备坐标帧的0度面向前（沿着X轴方向）
•  Header header
•  float32 angle_min        # scan的开始角度 [弧度]
•  float32 angle_max        # scan的结束角度 [弧度]
•  float32 angle_increment  # 测量的角度间的距离 [弧度]
•  float32 time_increment   # 测量间的时间 [秒]
•  float32 scan_time        # 扫描间的时间 [秒]
•  float32 range_min        # 最小的测量距离 [米]
•  float32 range_max        # 最大的测量距离 [米]
•  float32[] ranges         # 测量的距离数据 [米] (注意：值 <
   range_min 或 > range_max 应当被丢弃)
•  float32[] intensities    # 强度数据 [device-specific units]
```

# Obstacle avoidance for robots

TurtleBot™

## step 1.Creating a catkin Package

```
$cd ~/smartcar_ws/src
$catkin_create_pkg smartcar_demo actionlib actionlib_msgs geometry_msgs interactive_markers nav_msgs rospy
sensor_msgs std_msgs visualization_msgs
```

## step 2.Write code to avoid obstacles

```
$cd  smartcar_demo
$mkdir scripts
$cd scripts
$touch smartcar_obstacle.py
```

# Obstacle avoidance for robo

Bot™

smartcar_obstacle

src > smartcar_demo > scripts > smartcar_obstacle > ...

```python
import rospy
import math
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

LINEAR_VEL = 0.22
STOP_DISTANCE = 0.2
LIDAR_ERROR = 0.05
SAFE_STOP_DISTANCE = STOP_DISTANCE + LIDAR_ERROR

class Obstacle():
    def __init__(self):
        self._cmd_pub = rospy.Publisher('cmd_vel', Twist, queue_size=1)
        self.obstacle()

    def get_scan(self):
        scan = rospy.wait_for_message('scan', LaserScan)
        scan_filter = []

        samples = len(scan.ranges)  # The number of samples is defined in
                                    # turtlebot3_<model>.gazebo.xacro file,
                                    # the default is 360.
        samples_view = 1            # 1 <= samples_view <= samples

        if samples_view > samples:
            samples_view = samples

        if samples_view is 1:
            scan_filter.append(scan.ranges[0])

        else:
            left_lidar_samples_ranges = -(samples_view//2 + samples_view % 2)
            right_lidar_samples_ranges = samples_view//2

            left_lidar_samples = scan.ranges[left_lidar_samples_ranges:]
            right_lidar_samples = scan.ranges[:right_lidar_samples_ranges]
            scan_filter.extend(left_lidar_samples + right_lidar_samples)

        for i in range(samples_view):
            if scan_filter[i] == float('Inf'):
                scan_filter[i] = 3.5
            elif math.isnan(scan_filter[i]):
                scan_filter[i] = 0

        return scan_filter

    def obstacle(self):
        twist = Twist()
        turtlebot_moving = True

        while not rospy.is_shutdown():
            lidar_distances = self.get_scan()
            min_distance = min(lidar_distances)

            if min_distance < SAFE_STOP_DISTANCE:
                if turtlebot_moving:
                    twist.linear.x = 0.0
                    twist.angular.z = 0.0
                    self._cmd_pub.publish(twist)
                    turtlebot_moving = False
                    rospy.loginfo('Stop!')
            else:
                twist.linear.x = LINEAR_VEL
                twist.angular.z = 0.0
                self._cmd_pub.publish(twist)
                turtlebot_moving = True
                rospy.loginfo('Distance of the obstacle : %f', min_distance)

def main():
    rospy.init_node('smartcar_obstacle')
    try:
        obstacle = Obstacle()
    except rospy.ROSInterruptException:
        pass

if __name__ == '__main__':
    main()
```

# Obstacle avoidance for robots

TurtleBot™

1. In terminal

```
$roslaunch smartcar_gazebo smartcar_with_laser_nav.launch
```

2. In new terminal

```
$rosrun smartcar_demo smartcar_obstacle.py
```

lab task1:自己实现并截效果图，写入报告。注意这里若需要好的效果，需要多调试修改参数。

# Smartcar explore maze

$cd ~/smartcar_ws/src/**smartcar_demo/**scripts
$touch maze_explorer.py

maze_explorer.py 6

src > smartcar_teleop > scripts > maze_explorer.py > ...

```python
#!/usr/bin/env python

import rospy, time
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
def scan_callback(msg):
    global range_front
    global range_right
    global range_left
    global ranges
    global min_front,i_front, min_right,i_right, min_left ,i_left
    ranges = msg.ranges
    # get the range of a few points
    # in front of the robot (between 5 to -5 degrees)
    range_front[:5] = msg.ranges[5:0:-1]
    range_front[5:] = msg.ranges[-1:-5:-1]
    # to the right (between 300 to 345 degrees)
    range_right = msg.ranges[300:345]
    # to the left (between 15 to 60 degrees)
    range_left = msg.ranges[60:15:-1]
    # get the minimum values of each range
    # minimum value means the shortest obstacle from the robot
    min_range,i_range = min( (ranges[i_range],i_range) for i_range in xrange(len(ranges)) )
    min_front,i_front = min( (range_front[i_front],i_front) for i_front in xrange(len(range_front)) )
    min_right,i_right = min( (range_right[i_right],i_right) for i_right in xrange(len(range_right)) )
    min_left ,i_left  = min( (range_left [i_left ],i_left ) for i_left  in xrange(len(range_left )) )

# Initialize all variables
range_front = []
range_right = []
range_left  = []
min_front = 0
i_front = 0
min_right = 0
i_right = 0
min_left = 0
i_left = 0
# Create the node
cmd_vel_pub = rospy.Publisher('cmd_vel', Twist, queue_size = 1) # to move the robot
scan_sub = rospy.Subscriber('scan', LaserScan, scan_callback)   # to read the laser scanner
rospy.init_node('maze_explorer')

command = Twist()
command.linear.x = 0.0
command.angular.z = 0.0
rate = rospy.Rate(10)
time.sleep(1) # wait for node to initialize

near_wall = 0 # start with 0, when we get to a wall, change to 1
# Turn the robot right at the start
# to avoid the 'looping wall'
print("Turning...")
command.angular.z = -0.5
command.linear.x = 0.1
cmd_vel_pub.publish(command)
time.sleep(2)
while not rospy.is_shutdown():
    # The algorithm:
    # 1. Robot moves forward to be close to a wall
    # 2. Start following left wall.
    # 3. If too close to the left wall, reverse a bit to get away
    # 4. Otherwise, follow wall by zig-zagging along the wall
    # 5. If front is close to a wall, turn until clear
    while(near_wall == 0 and not rospy.is_shutdown()): #1
        print("Moving towards a wall.")
        if(min_front > 0.2 and min_right > 0.2 and min_left > 0.2):
            command.angular.z = -0.1    # if nothing near, go forward
            command.linear.x = 0.15
            print "C"
        elif(min_left < 0.2):           # if wall on left, start tracking
            near_wall = 1
            print "A"
        else:
            command.angular.z = -0.25   # if not on left, turn right
            command.linear.x = 0.0

        cmd_vel_pub.publish(command)
    else:   # left wall detected
        if(min_front > 1.5): #2
            if(min_left < 0.5):    #3
                print("Range: {:.2f}m - Too close. Backing up.".format(min_left))
                command.angular.z = -1.2
                command.linear.x = -0.1
            elif(min_left > 0.8):  #4
                print("Range: {:.2f}m - Wall-following; turn left.".format(min_left))
                command.angular.z = 1.2
                command.linear.x = 0.15
            else:
                print("Range: {:.2f}m - Wall-following; turn right.".format(min_left))
                command.angular.z = -1.2
                command.linear.x = 0.15
        else:   #5
            print("Front obstacle detected. Turning away.")
            command.angular.z = -1.0
            command.linear.x = 0.0
            cmd_vel_pub.publish(command)
            while(min_front < 0.3 and not rospy.is_shutdown()):
                pass
        # publish command
        cmd_vel_pub.publish(command)
    # wait for the loop
    rate.sleep()
```

# Obstacle avoidance for robots

1. In terminal

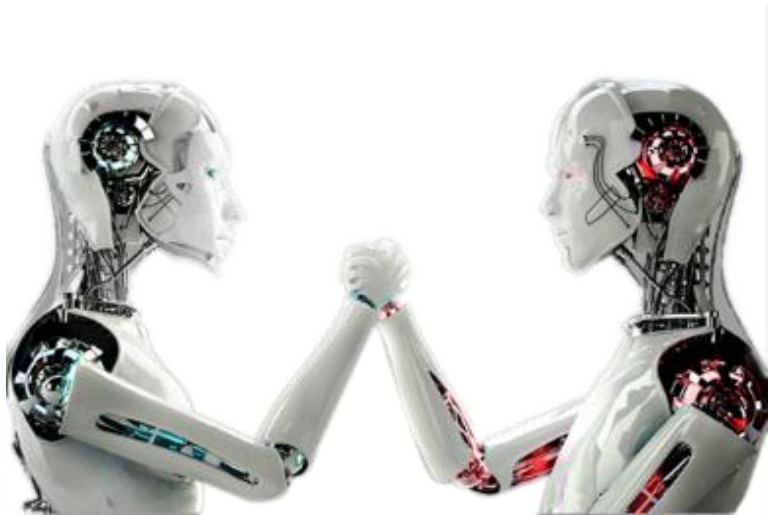
```
$roslaunch smartcar_gazebo smartcar_with_laser_nav.launch
```

2. In new terminal

```
$rosrun smartcar_demo maze_explorer.py
```

lab task2:自己实现并截效果图，写入报告。注意这里若需要好的效果，需要多调试修改参数。

# ROS+OPENCV



# ROS+Opencv

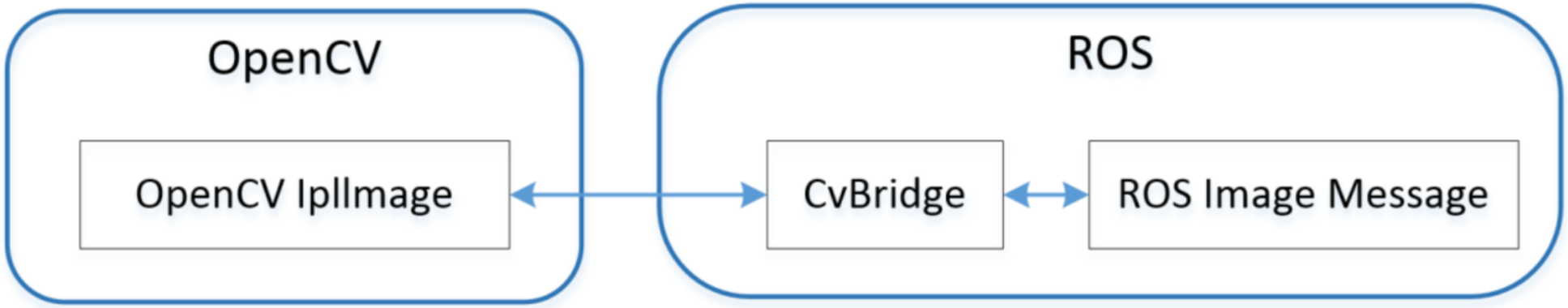
TurtleBot™

- Open Source Computer Vision Library
- Supports Windows, Linux, Android and Mac OS
- Has C++, Python, Java and MATLAB interfaces
- Has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

OpenCV

# ROS+Opencv

ROS passes around images in its own **sensor_msgs/Image** message format, but many users will want to use images in conjunction with OpenCV. **CvBridge is a ROS library that provides an interface between ROS and OpenCV**




## ➢ Install Opencv

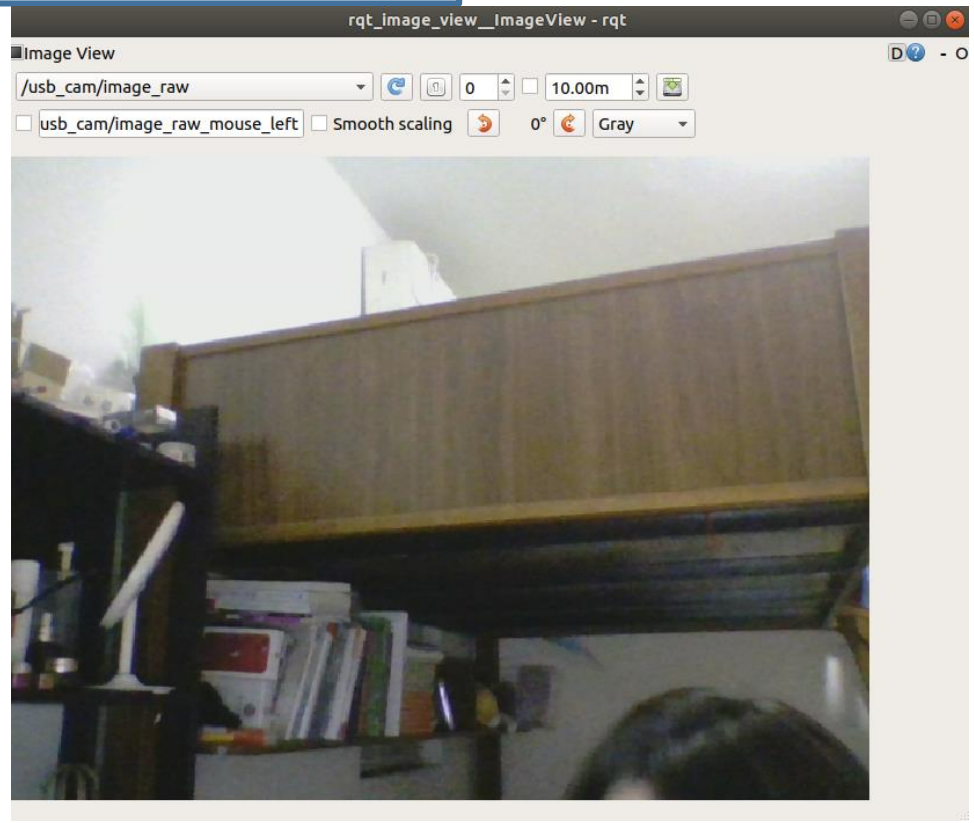
```
$sudo apt-get install ros-melodic-vision-opencv libopencv-dev python-opencv
```

# ROS+Opencv

## ➢A simple example

```
$sudo apt-get install ros-melodic-usb-cam    # 安装摄像头功能包
$roslaunch usb_cam usb_cam-test.launch       # 启动功能包
$rqt_image_view                              # 可视化工具
```

```
(base) jyh@jyh:~/Downloads$ rostopic list
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/rosout
/rosout_agg
/usb_cam/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descriptions
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
(base) jyh@jyh:~/Downloads$
```



# ROS+Opencv



**Step1:** In your **package.xml** and **CMakeLists.txt** (or when you use **catkin_create_pkg**), add the following dependencies:

```
sensor_msgs
cv_bridge
roscpp
std_msgs
image_transport
```

```
$cd ~/catkin_ws/src
$catkin_create_pkg learning_opencv sensor_msgs cv_bridge
roscpp std_msgs image_transport
```

**step2:** Modify CMakeLists.txt

Add one line in CMakeLists.txt:

```
find_package(OpenCV REQUIRED)
```

# ROS+Opencv

**step3:** Create a node(Write a cpp file)

image_converter.cpp

src > learning_opencv > src > image_converter.cpp > OPENCV_WINDOW

```cpp
//
// Created by jyh on 2020/4/6.
//
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>

using namespace std;
using namespace cv;
static const std::string OPENCV_WINDOW = "Image window";

class ImageConverter
{
    ros::NodeHandle nh_;
    image_transport::ImageTransport it_;
    image_transport::Subscriber image_sub_;
    image_transport::Publisher image_pub_;

public:
    ImageConverter()
        : it_(nh_)
    {
        // Subscrive to input video feed and publish output video feed
        image_sub_ = it_.subscribe("/usb_cam/image_raw", 1,
                                   &ImageConverter::imageCb, this);
        image_pub_ = it_.advertise("/image_converter/output_video", 1);

        cv::namedWindow(OPENCV_WINDOW);
    }

    ~ImageConverter()
    {
        cv::destroyWindow(OPENCV_WINDOW);
    }

    void imageCb(const sensor_msgs::ImageConstPtr& msg)
    {
        cv_bridge::CvImagePtr cv_ptr;
        try
        {
            cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
        }
        catch (cv_bridge::Exception& e)
        {
            ROS_ERROR("cv_bridge exception: %s", e.what());
            return;
        }

        // Draw an example circle on the video stream
        if (cv_ptr->image.rows > 60 && cv_ptr->image.cols > 60)
            cv::circle(cv_ptr->image, cv::Point(50, 50), 10, CV_RGB(255,0,0));



        // Update GUI Window
        cv::imshow(OPENCV_WINDOW, cv_ptr->image);
        cv::waitKey(3);

        // Output modified video stream
        image_pub_.publish(cv_ptr->toImageMsg());
    }
};

int main(int argc, char** argv)
{
    ros::init(argc, argv, "image_converter");
    ImageConverter ic;
    ros::spin();
    return 0;
}
```

# ROS+Opencv

**step4:** Building node

Add two lines to the bottom of the **CMakeLists.txt:**

```
add_executable(image_converter src/image_converter.cpp)
target_link_libraries(image_converter ${catkin_LIBRARIES} ${OpenCV_LIBRARIES})
```

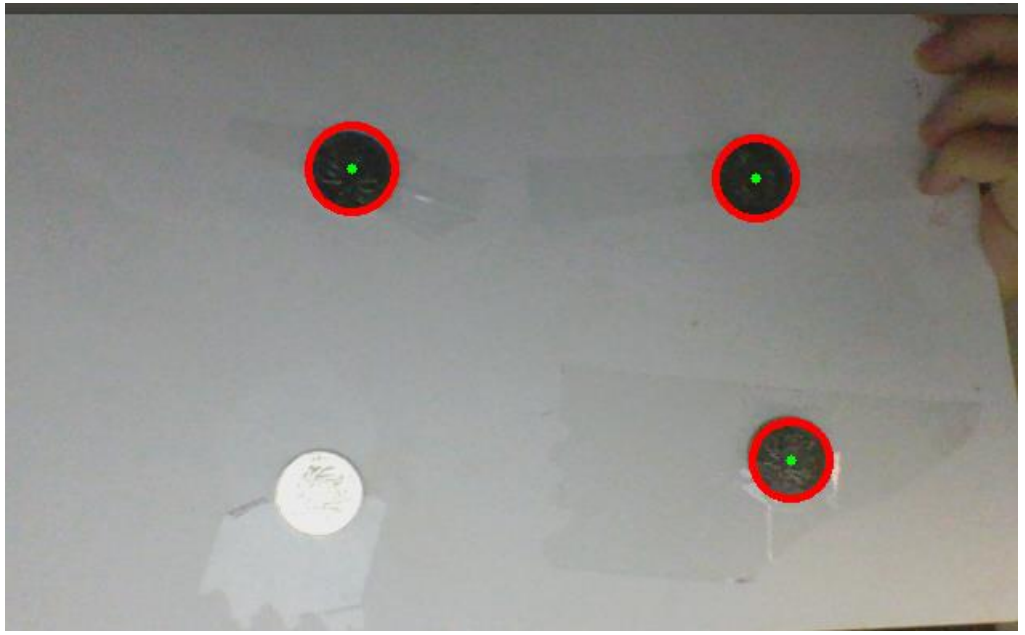
**step5:** Run to see the results

```
$source devel/setup.bash
$rosrun learning_opencv image_converter
```

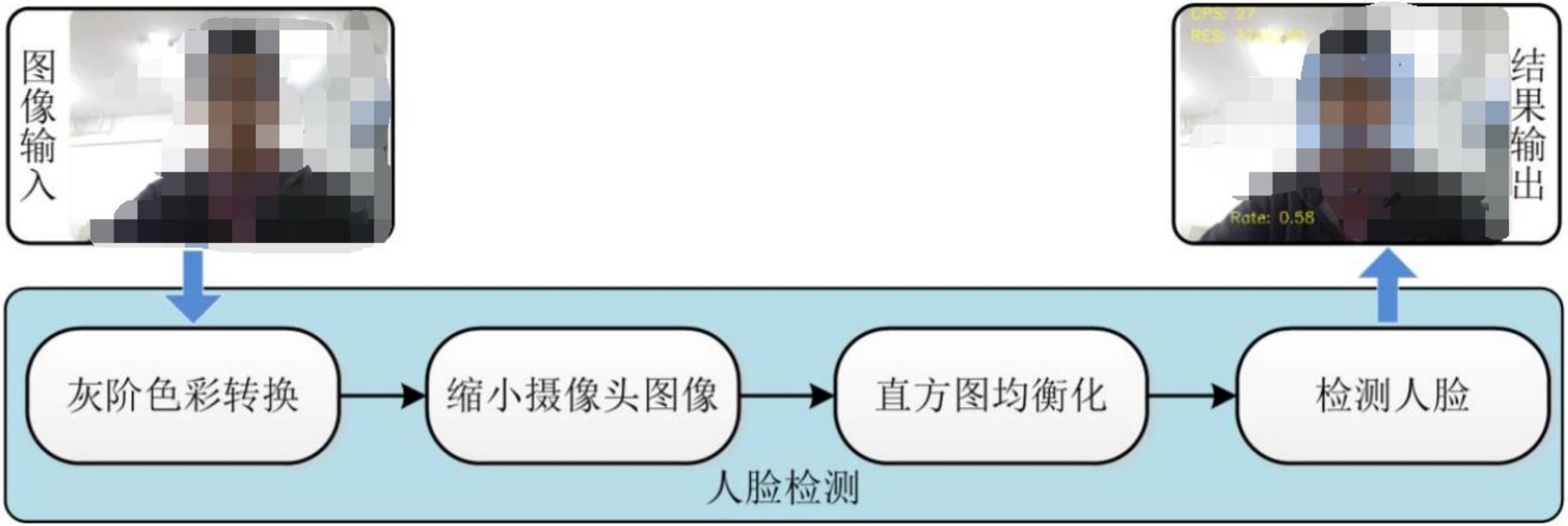

```
(base) jyh@jyh:/media/jyh/FILE/2022SP/CS401/LAB_SRC/catkin_ws$ rostopic list
/image_converter/output_video
/image_converter/output_video/compressed
/image_converter/output_video/compressed/parameter_descriptions
/image_converter/output_video/compressed/parameter_updates
/image_converter/output_video/compressedDepth
/image_converter/output_video/compressedDepth/parameter_descriptions
/image_converter/output_video/compressedDepth/parameter_updates
/image_converter/output_video/theora
/image_converter/output_video/theora/parameter_descriptions
/image_converter/output_video/theora/parameter_updates
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/rosout
/rosout_agg
/usb_cam/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descriptions
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
(base) jyh@jyh:/media/jyh/FILE/2022SP/CS401/LAB_SRC/catkin_ws$
```

# ➢Detecting Circles With OpenCV

```
void imageCb(const sensor_msgs::ImageConstPtr& msg)
{
    cv_bridge::CvImagePtr cv_ptr;
    try
    {
        cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
    }
    catch (cv_bridge::Exception& e)
    {
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }

    //detect circles
    Mat img_cvt;
    cvtColor(cv_ptr->image, img_cvt, CV_BGR2GRAY);
    threshold(img_cvt, img_cvt, 0, 255, CV_THRESH_BINARY + CV_THRESH_OTSU);
    //medianBlur(img_cvt, img_cvt, 3);
    GaussianBlur(img_cvt, img_cvt, Size(9, 9), 2,2);
    vector<Vec3f> circles;
    HoughCircles(img_cvt, circles, CV_HOUGH_GRADIENT, 1, 50, 100, 15, 20, 50);

    for (int  i = 0; i < circles.size(); i++)
    {
        Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
        int    R = cvRound(circles[i][2]);
        circle(cv_ptr->image, center, 3, Scalar(0, 255, 0), -1, 8, 0);
        circle(cv_ptr->image, center, R, Scalar(0, 0, 255), 3, 8, 0);
    }
    // Update GUI Window
    cv::imshow(OPENCV_WINDOW, cv_ptr->image);
    cv::waitKey(3);

    // Output modified video stream
    image_pub_.publish(cv_ptr->toImageMsg());
}
```



## ➢Detecting faces




**基于Haar特征的级联分类器对象检测算法**

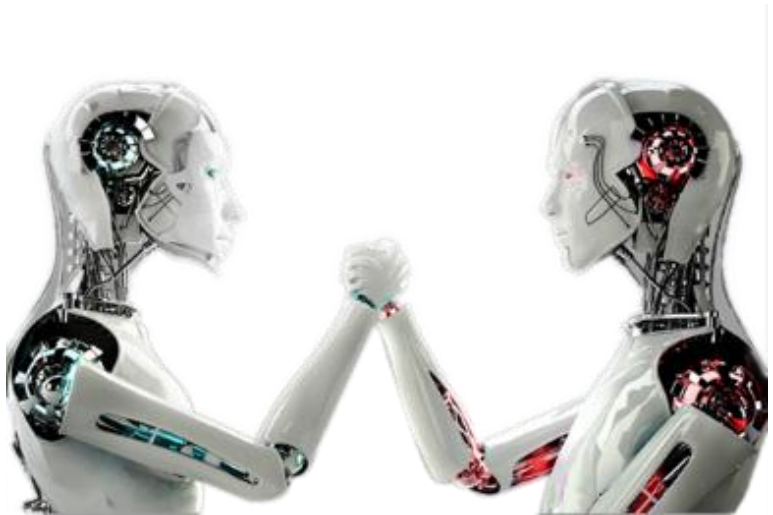
# ROS+Opencv



## ➢Detecting faces

```
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
            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
```

# YOLO ROS

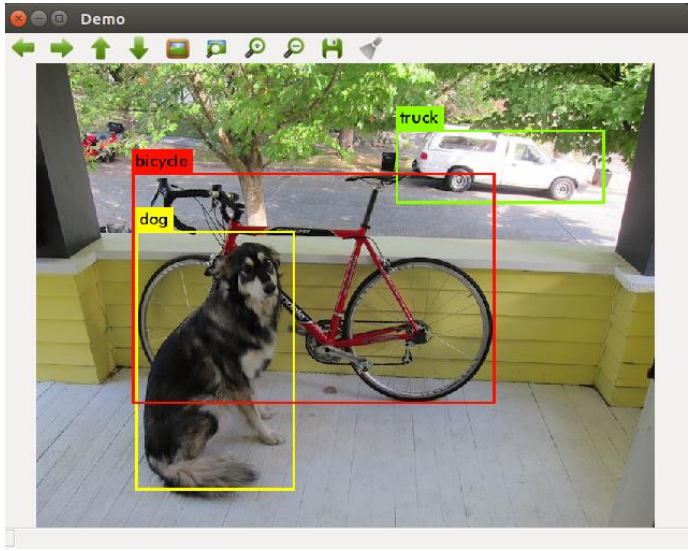


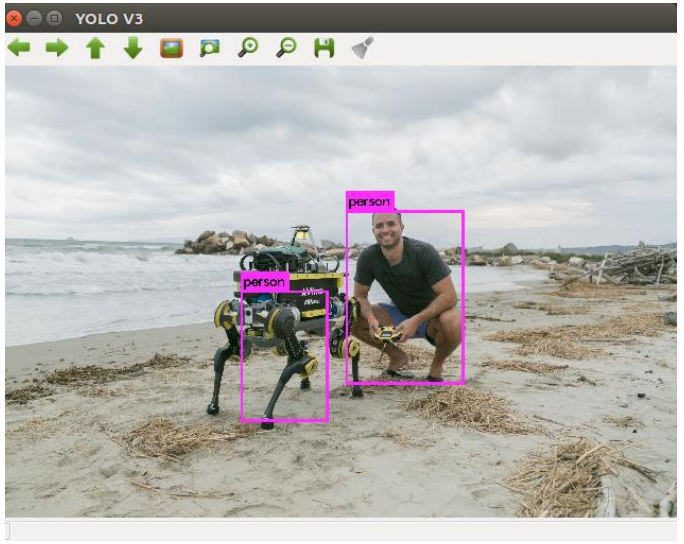
# YOLO introduction

For more details :
https://pjreddie.com/darknet/yolo/

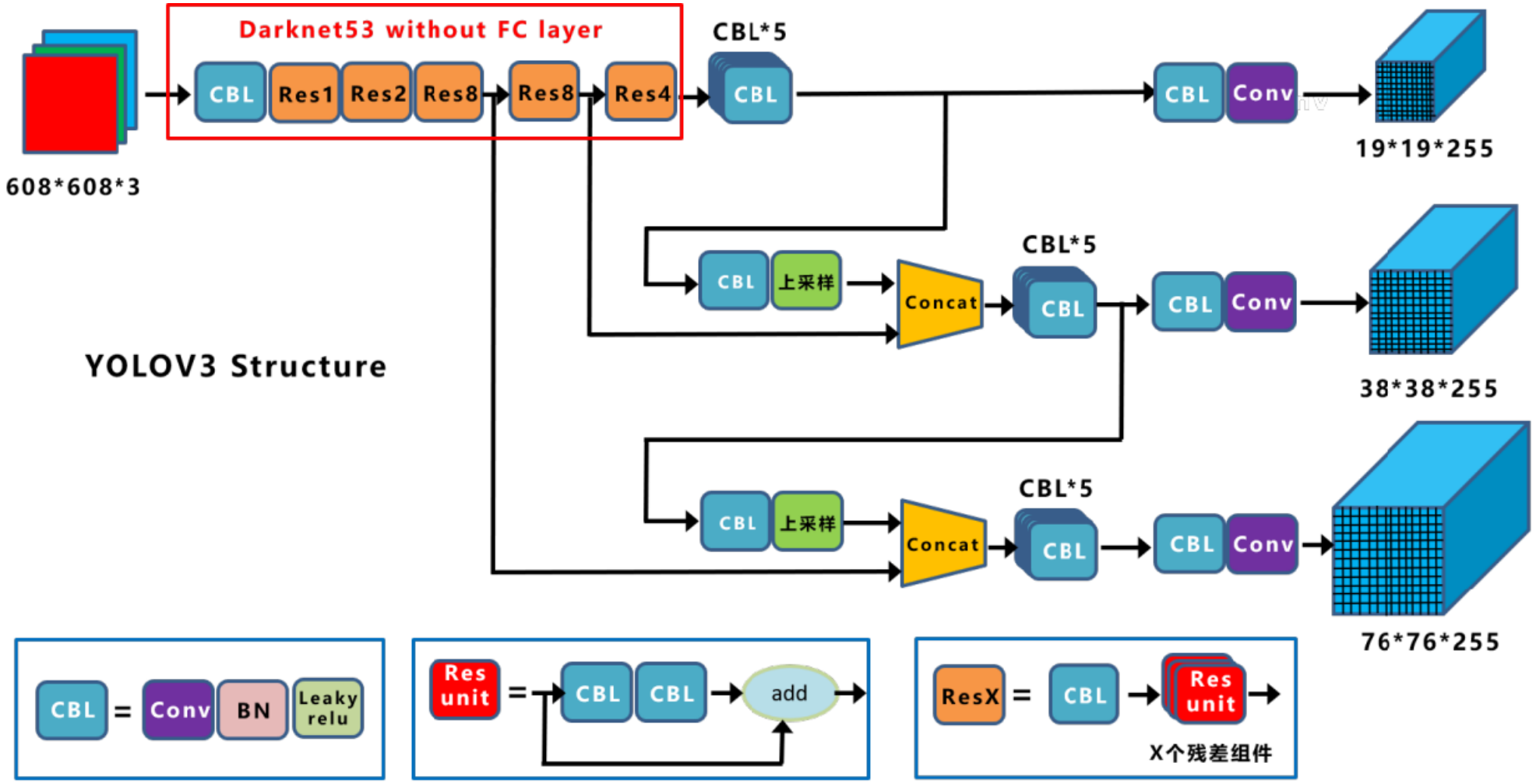
TurtleBot™

- YOLO（You Only Look Once）是2016年提出的一篇关于目标检测比较有名的文章。其以速度快和泛化能力强为优点。在YOLO之后，又改进出了YOLO-v2、YOLO-v3(2018). v2、v3的精度相比v1有了很大的提升.
- 2020年初YOLO之父Joseph Redmon宣布退出计算机视觉的研究。
- 在原作者声名放弃更新Yolo算法后，俄罗斯的Alexey大神在同年4月23日推出YOLO V4，其在COCO数据集上的平均精度(AP)和帧率精度(FPS)分别提高了10% 和12%，并得到了Joseph Redmon的官方认可，被认为是当前最强的实时对象检测模型之一。
- 同年6月25日，Ultralytics发布了YOLOV5，其性能与YOLO V4不相伯仲，同样也是现今最先进的对象检测技术，并在推理速度上是目前最强。。
- 而到了2021年，就在大家质疑Yolo系列该如何改进时？旷视科技又发布了Yolox算法。



# YOLOv3 Structure




https://blog.csdn.net/nan355655600/article/details/106246625

# The use of YOLO in ROS (YOLO ROS)

step1: Download code

```
cd ~/smartcar_ws/src
git clone --recursive git@github.com:leggedrobotics/darknet_ros.git
```

step2: Download weights

```
cd ~/smartcar_ws/src/darknet_ros/darknet_ros/yolo_network_config/weights/
wget http://pjreddie.com/media/files/yolov2.weights
wget http://pjreddie.com/media/files/yolov2-tiny.weights
wget https://pjreddie.com/media/files/yolov3.weights
wget https://pjreddie.com/media/files/yolov3-tiny.weights
```

step3: Building

```
cd ~/smartcar_ws
catkin_make -DCMAKE_BUILD_TYPE=Release
```

step4: Test

① 发布图像话题

直接将电脑自带摄像头或连接电脑的USB摄像头采集的图像发布为ROS图像话题并使用。

```
sudo apt-get install ros-kinetic-usb-cam
roslaunch usb_cam usb_cam-test.launch
```

step4: Test

② 运行darknet_ros

执行darknet_ros进行检测，在运行检测之前需要更改一下文件，使得darknet_ros订阅的话题与usb_cam发布的图片话题对应。打开 darknet_ros/config/ros.yaml文件，找到：

```
jyh@jyh:~/smartcar_ws$ rostopic list
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/rosout
/rosout_agg
/usb_cam/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descriptio
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
jyh@jyh:~/smartcar_ws$
```

rc > darknet_ros > darknet_ros > config > ! ros.yaml

```yaml
subscribers:

  camera_reading:
    topic: /camera/rgb/image_raw
    queue_size: 1

actions:

  camera_reading:
    name: /darknet_ros/check_for_objects

publishers:

  object_detector:
    topic: /darknet_ros/found_object
    queue_size: 1
    latch: false
```

src > darknet_ros > darknet_ros > config > ! ros.yaml

```yaml
subscribers:

  camera_reading:
    topic: /usb_cam/image_raw
    queue_size: 1

actions:

  camera_reading:
    name: /darknet_ros/check_for_objects

publishers:

  object_detector:
    topic: /darknet_ros/found_object
    queue_size: 1
    latch: false

  bounding_boxes:
    topic: /darknet_ros/bounding_boxes
    queue_size: 1
    latch: false
```

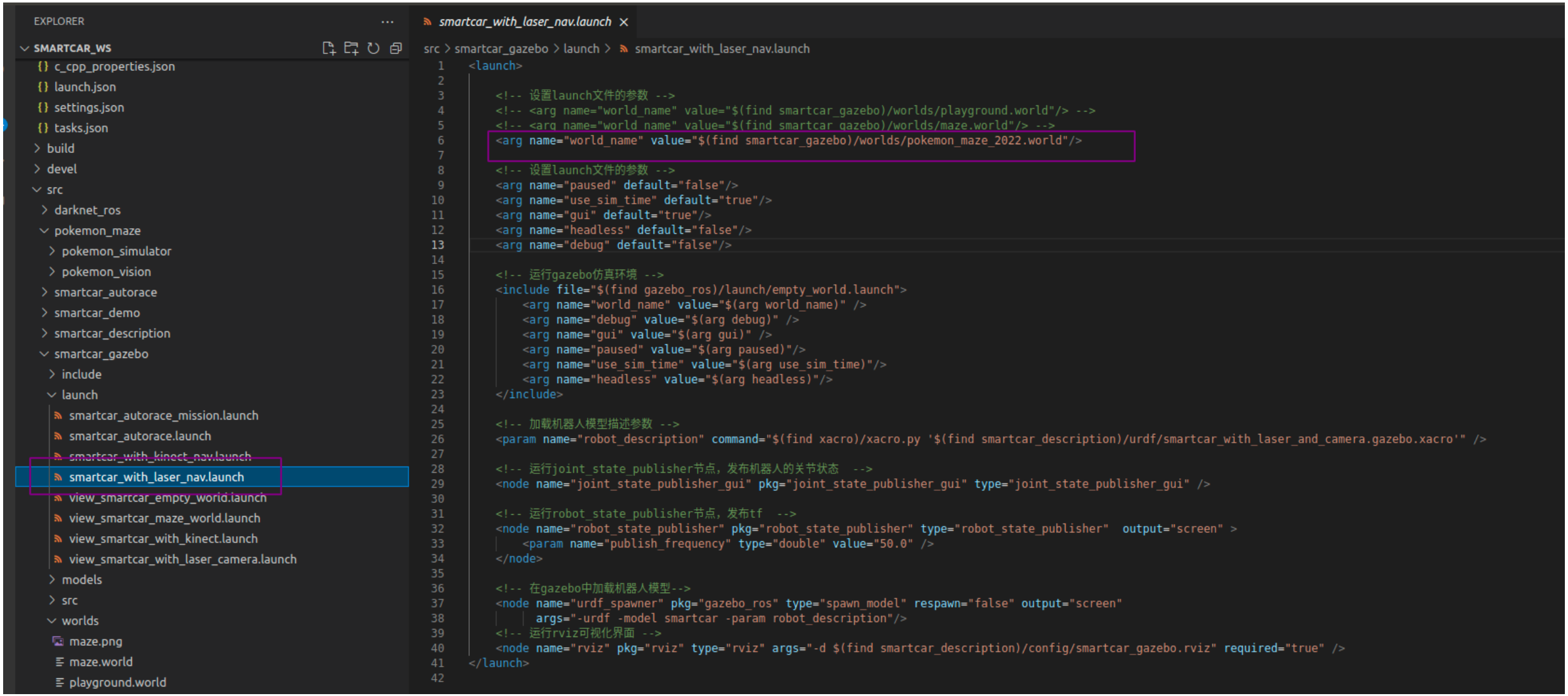
# YOLO ROS

③ 回到smartcar_ws目录，执行：

```
source devel/setup.bash
roslaunch darknet_ros darknet_ros.launch
```

# YOLO for smartcar

TurtleBot™



# YOLO for smartcar

- 根据smartcar图像话题修改ros.yaml

```
(base) jyh@jyh:/media/jyh/FILE/2022SP/CS401/LAB_SRC$ rostopic list
/camera/camera_info
/camera/image_raw
/camera/image_raw/compressed
/camera/image_raw/compressed/parameter_descriptions
/camera/image_raw/compressed/parameter_updates
/camera/image_raw/compressedDepth
/camera/image_raw/compressedDepth/parameter_descriptions
/camera/image_raw/compressedDepth/parameter_updates
/camera/image_raw/theora
/camera/image_raw/theora/parameter_descriptions
/camera/image_raw/theora/parameter_updates
/camera/parameter_descriptions
/camera/parameter_updates
/clicked_point
/clock
```

src > darknet_ros > darknet_ros > config > ! ros.yaml

```yaml
subscribers:

  camera_reading:
    topic: /camera/image_raw
    queue_size: 1

actions:

  camera_reading:
    name: /darknet_ros/check_for_objects

publishers:

  object_detector:
    topic: /darknet_ros/found_object
    queue_size: 1
    latch: false
```

# YOLO for smartcar

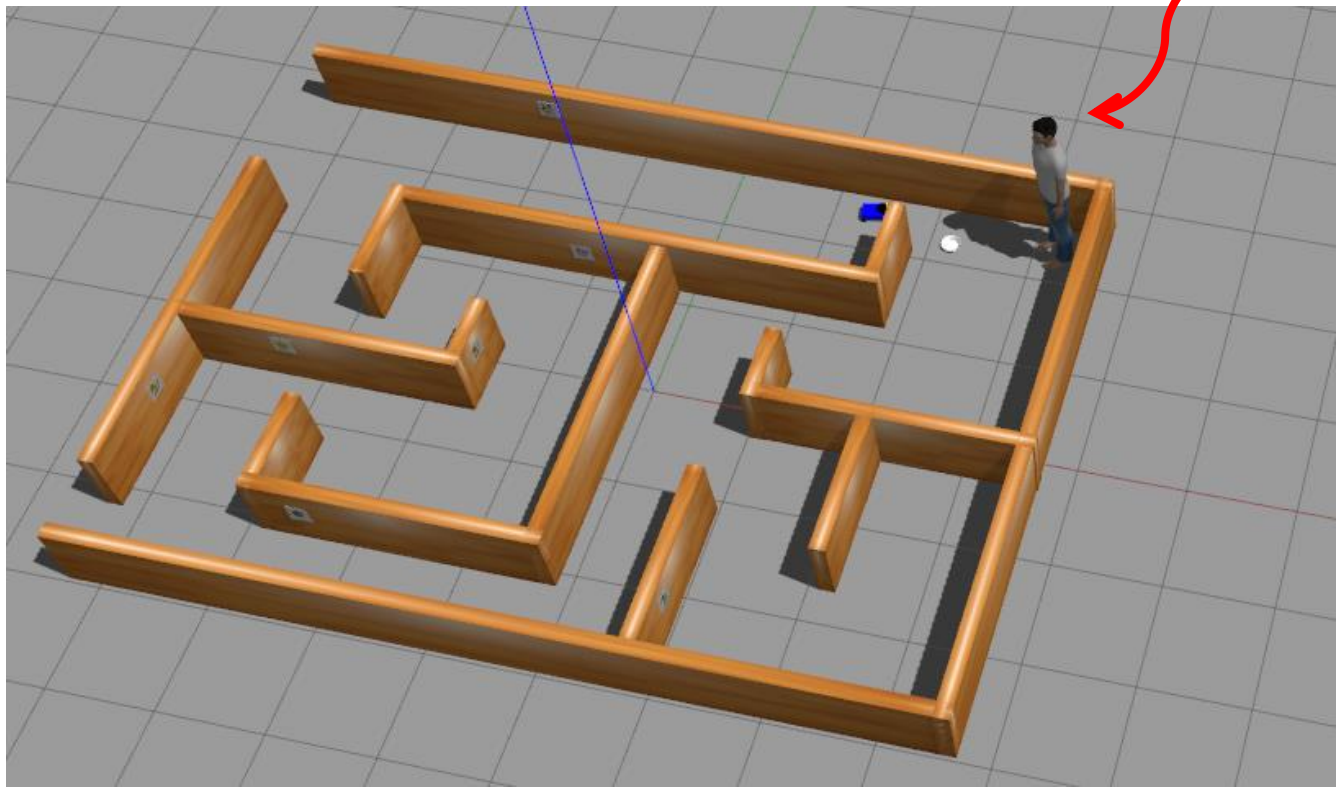
回到smartcar_ws目录，执行：

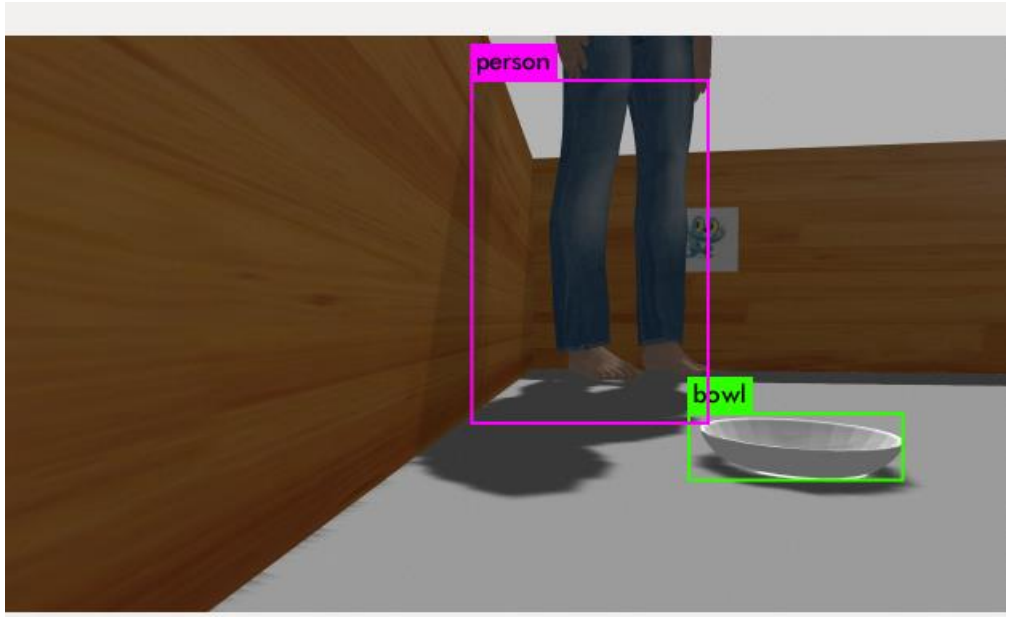
```
source devel/setup.bash
roslaunch smartcar_gazebo smartcar_with_laser_nav.launch
```

另开teminal:

```
source devel/setup.bash
roslaunch darknet_ros darknet_ros.launch
```

*在gazebo中加入person和一个bowl*





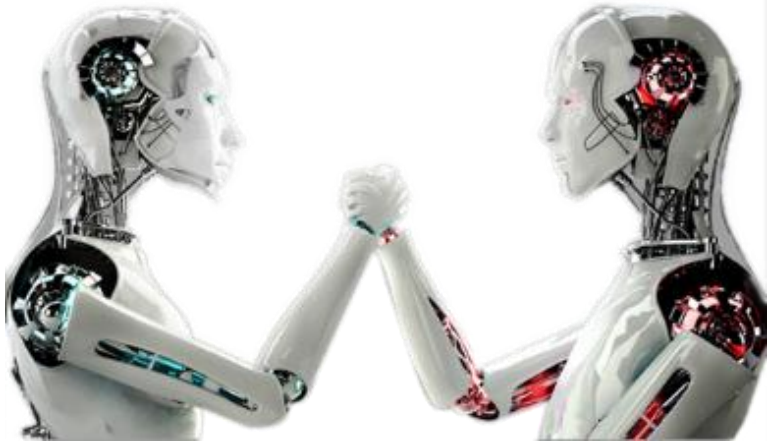
# YOLO for smartcar

TurtleBot™

lab task3:There are lots of Pokemon on the walls of the maze and your task is to detect them.
Note: the maze environment is given by project.

PART FOUR

# Lab Task



# Lab Tasks

**This lab contains three tasks that need to be completed. When you're done, you need to submit a ZIP file and a report to the BB.**

- **The ZIP file is the compressed file of smartcar_Demo package.**
- **Lab tasks 1, 2, and 3 are all written in this one report.**
- **The report contains code, running steps, running results, results analysis, and other contents you want to share!**

# Lab References

## Futher References

- ROS cv_bridge
  http://wiki.ros.org/cv_bridge
- ROS opencv_apps
  http://wiki.ros.org/opencv_apps
- Opencv tutorials
  https://docs.opencv.org/2.4/doc/tutorials/tutorials.html

# Thanks