Vivek Ponnala

Object Detection Robot

Introduction

Overview

In this automation world, as the technology is being updated day by day, object detection is the main ability for robots to interact with an environment and perform tasks correctly. In the field of this robotics active recognition, a robot is integrated with a camera, and raw image data is processed to recognize objects correctly. Here, we have used an object recognition model to detect objects and necessary steps are taken. This robot can be deployed in many places like offices, larger companies, and different other places where the detection of objects is the main task. In this project, you will create one such robot application in the Gazebo simulator and perform the object detection user verification tasks using ROS and OpenCV.

Purpose

You'll be able to work with the most powerful open-source robotics framework i.e.
ROS(Robot Operating System),
You'll be in a position to create ROS packages for Robot applications and Simulation
applications,
Building a robot that can move autonomously in the Gazebo simulator by keyboard
control
Simulating the robot on Gazebo simulator by considering real environment parameters
and Detecting and Displaying the name of the detected objects inside the Gazebo
simulator with python and OpenCV.
Detecting objects from raw image data from the robot with the YOLO Object detection
model.

Literature Survey

Existing problem

First, they may not have enough data-points for that object type or that specific object. Second, online images may be available for some canonical views, but in practice a robot may see the object from any ar- bitrary view. Objects can appear drastically different from different viewing angles.

Proposed Solution

The complete application that you will develop is a robot application in the simulation world integrated with vision and object detection capabilities. The robot will fetch image data from the camera and process the image and detects the object in that image from the simulation environment and perform specified actions depending on our environment. To accomplish this project you use the ROS Melodic version, python OpenCV, YOLO Object detection model to detect objects with Visual studio code as IDE to develop this application.

Theoretical Analysis

Block Diagram

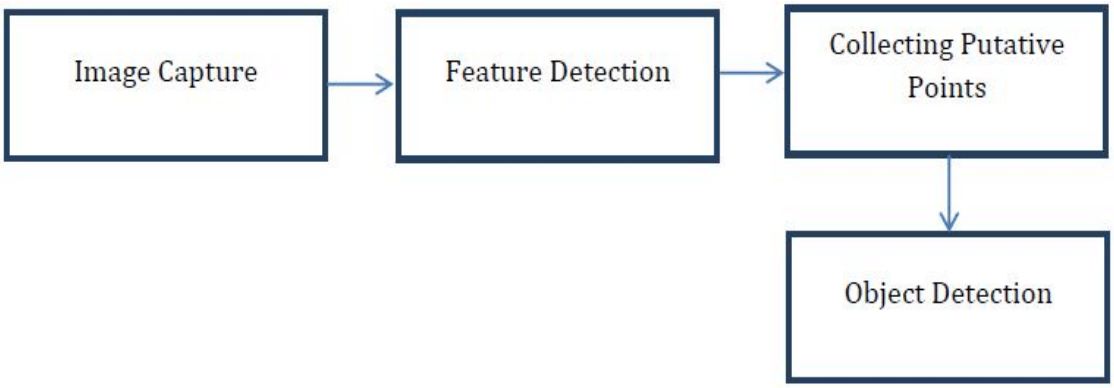


Figure 1: Block Diagram

Hardware / Software designing

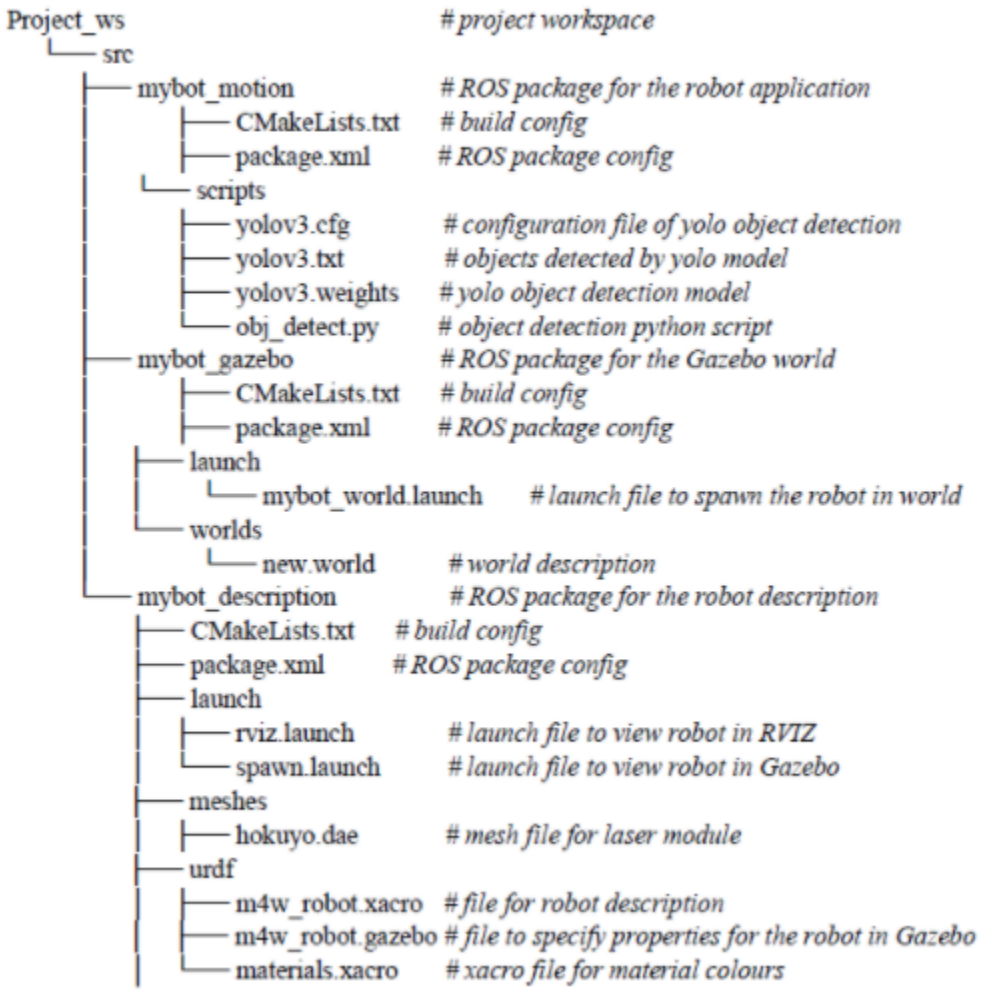


Figure 2: Software design

Experimental Investigations

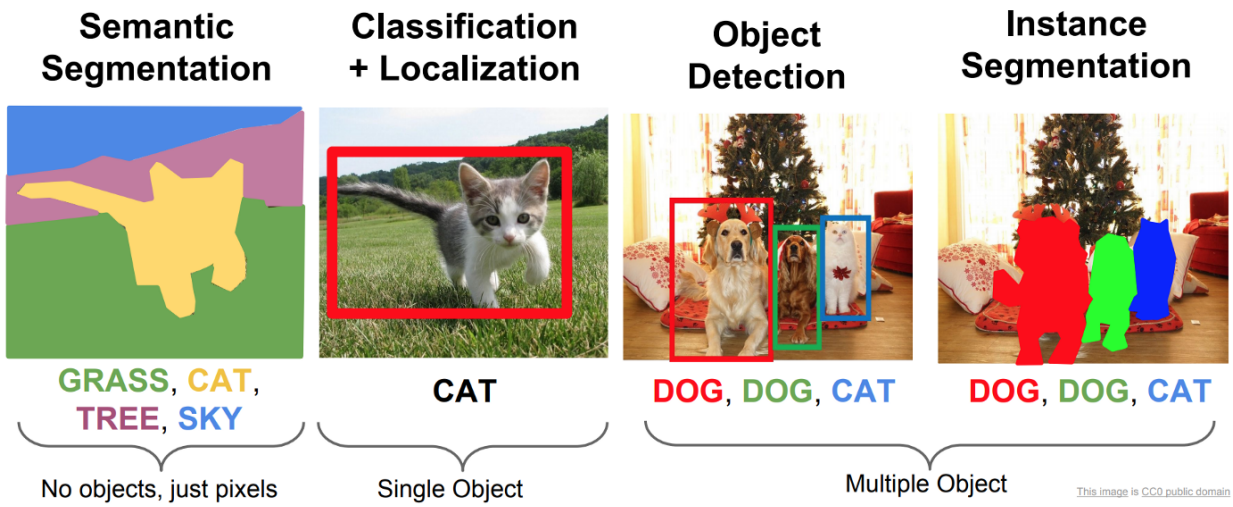


Figure 3: Experimental investigations

Flowchart

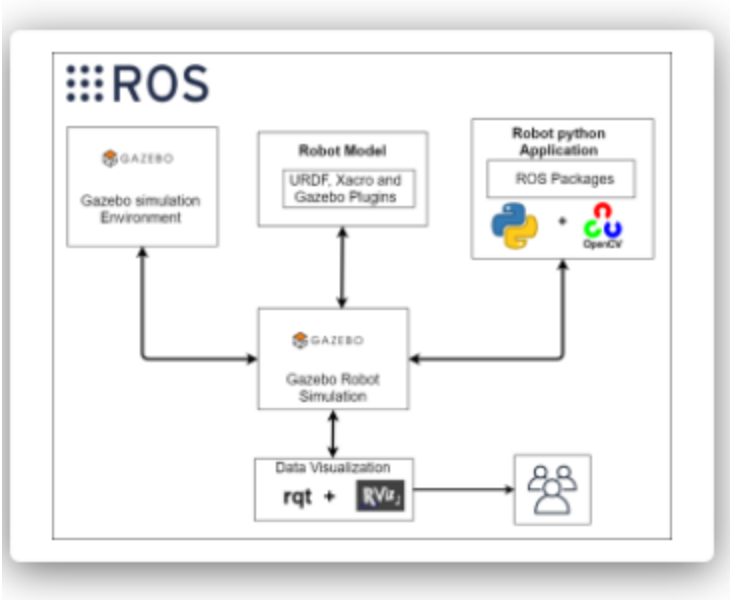


Figure 4: Flow-chart for object detection robot

Result

Gazebo simulator loads with cone, stop sign, traffic lights, car, fire hose and person

With raw_image/camera topic rqt shows cone, stop sign, traffic lights, car, fire hose and person

After running obj_detect.py, objects are successfully detected

Advantages & Disadvantages

Real-time detection

Simple network structure

Low detection precision

Locate objects with horizontal bounding box

Poor results for small and dense objects

Easy to mislocate

Applications

Object detection is applied in numerous territories of image processing, including picture retrieval, security, observation, computerized vehicle systems and machine investigation.

Conclusion

Object detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning.

Future Scope

The future of object detection technology is in the process of proving itself, and much

like the original Industrial Revolution, it has the potential to free people from menial jobs that can be done more efficiently and effectively by machines. It will also open up new avenues of research and operations that will reap additional benefits in the future.

Bibliography

https://smartinternz.com/Student/guided_project_info/5077#

https://www.ijcai.org/Proceedings/11/Papers/346.pdf

https://www.researchgate.net/figure/Block-Diagram-for-Object-Detection_fig1_310769942

https://www.researchgate.net/figure/The-advantages-and-disadvantages-of-existing-object-detection-methods_tbl1_338847053

https://www.pixelsolutionz.com/application-object-detection-real-life/

https://blog.rebellionresearch.com/blog/the-future-of-object-detection

Appendix

Source code

```
<?xml version="1.0"?>
```

```
<launch>

  <param name="robot_description" command="$(find xacro)/xacro.py '$(find
mybot_description)/urdf/m4w_robot.xacro'"/>

  <!-- send fake joint values -->
  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher">
    <param name="use_gui" value="False"/>
  </node>

  <!-- Combine joint values -->
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher"/>

  <!-- Show in Rviz   -->
  <node name="rviz" pkg="rviz" type="rviz" />

</launch>

<?xml version="1.0" encoding="UTF-8"?>
<launch>
    <param name="robot_description" command="$(find xacro)/xacro.py '$(find
mybot_description)/urdf/m4w_robot.xacro'" />

    <arg name="x" default="0"/>
    <arg name="y" default="0"/>
    <arg name="z" default="0"/>

    <node name="mybot_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
          args="-urdf -param robot_description -model m2wr -x $(arg x) -y $(arg y) -z $(arg
z)" />
</launch>

<?xml version="1.0" ?>
<robot name="robot_1" xmlns:xacro="https://www.ros.org/wiki/xacro" >

<gazebo reference="base_link">
```

```
    <material>Gazebo/white</material>
  </gazebo>
  <gazebo reference="left_wheel">
    <material>Gazebo/Red</material>
  </gazebo>
  <gazebo reference="right_wheel">
    <material>Gazebo/Red</material>
  </gazebo>
<gazebo reference="left_f_wheel">
    <material>Gazebo/Orange</material>
  </gazebo>
<gazebo reference="right_f_wheel">
    <material>Gazebo/Orange</material>
  </gazebo>
  <gazebo reference="camera_link">
    <material>Gazebo/Red</material>
</gazebo>

  <!-- camera -->
  <gazebo reference="camera_link">
    <sensor type="camera" name="camera1">
      <update_rate>30.0</update_rate>
      <camera name="head">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>800</width>
          <height>800</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.02</near>
          <far>300</far>
        </clip>
      </camera>
      <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
```

```
      <cameraName>mybot/camera</cameraName>
      <imageTopicName>image_raw</imageTopicName>
      <cameraInfoTopicName>camera_info</cameraInfoTopicName>
      <frameName>camera_link</frameName>
      <hackBaseline>0.07</hackBaseline>
      <distortionK1>0.0</distortionK1>
      <distortionK2>0.0</distortionK2>
      <distortionK3>0.0</distortionK3>
      <distortionT1>0.0</distortionT1>
      <distortionT2>0.0</distortionT2>
    </plugin>
  </sensor>
 </gazebo>

<gazebo>
 <plugin name="skid_steer_drive_controller"
filename="libgazebo_ros_skid_steer_drive.so">
  <updateRate>100.0</updateRate>
  <robotNamespace>/</robotNamespace>
  <leftFrontJoint>left_f_wheel_joint</leftFrontJoint>
  <rightFrontJoint>right_f_wheel_joint</rightFrontJoint>
  <leftRearJoint>left_wheel_joint</leftRearJoint>
  <rightRearJoint>right_wheel_joint</rightRearJoint>
  <wheelSeparation>0.15</wheelSeparation>
  <wheelDiameter>0.07</wheelDiameter>
  <robotBaseFrame>base_link</robotBaseFrame>
  <torque>20</torque>
  <topicName>cmd_vel</topicName>
  <broadcastTF>false</broadcastTF>
 </plugin>
</gazebo>


 <!-- gazebo>
  <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">
   <alwaysOn>true</alwaysOn>
   <updateRate>20</updateRate>
```

```
    <leftJoint>left_f_wheel_joint</leftJoint>
    <rightJoint>right_f_wheel_joint</rightJoint>
    <wheelSeparation>0.15</wheelSeparation>
    <wheelDiameter>0.07</wheelDiameter>
    <torque>0.1</torque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>link_chassis</robotBaseFrame>
   </plugin>
  </gazebo -->

</robot>
<?xml version="1.0"?>
<robot name="m4w_robot" xmlns:xacro="http://www.ros.org/wiki/xacro">
<xacro:include filename="$(find mybot_description)/urdf/materials.xacro" />
<xacro:include filename="$(find mybot_description)/urdf/m4w_robot.gazebo" />

  <xacro:property name="base_width" value="0.16"/>
  <xacro:property name="base_len" value="0.2"/>
  <xacro:property name="wheel_radius" value="0.035"/>
  <xacro:property name="base_wheel_gap" value="0.007"/>
  <xacro:property name="wheel_separation" value="0.15"/>
  <xacro:property name="wheel_joint_offset" value="0.02"/>

  <xacro:macro name="box_inertia" params="m w h d">
   <inertial>
    <mass value="${m}"/>
    <inertia ixx="${m / 12.0 * (d*d + h*h)}" ixy="0.0" ixz="0.0" iyy="${m / 12.0 * (w*w +
h*h)}" iyz="0.0" izz="${m / 12.0 * (w*w + d*d)}"/>
   </inertial>
  </xacro:macro>

  <link name="base_footprint">
   <xacro:box_inertia m="20" w="0.001" h="0.001" d="0.001"/>
   <visual>
     <origin xyz="0 0 0" rpy="0 0 0" />
```

```
      <geometry>
        <box size="0.001 0.001 0.001" />
      </geometry>
        <material name="green"/>
   </visual>
  </link>

  <link name="base_link">
   <xacro:box_inertia m="10" w="${base_len}" h="${base_width}" d="0.02"/>
   <visual>
    <geometry>
     <box size="${base_len} ${base_width} 0.02"/>
    </geometry>
<material name="white"/>
   </visual>
   <collision>
    <geometry>
     <box size="${base_len} ${base_width} 0.02"/>
    </geometry>

   </collision>
  </link>
   <xacro:macro name="cylinder_inertia" params="m r h">
   <inertial>
    <mass value="${m}"/>
    <inertia ixx="${m*(3*r*r+h*h)/12}" ixy = "0" ixz = "0" iyy="${m*(3*r*r+h*h)/12}" iyz =
"0" izz="${m*r*r/2}"/>
   </inertial>
  </xacro:macro>

  <xacro:macro name="wheel" params="prefix reflect wheel_joint">
   <link name="${prefix}_wheel">
    <visual>
     <origin xyz="0 0 0" rpy="${pi/2} 0 0"/>
     <geometry>
      <cylinder radius="${wheel_radius}" length="0.01"/>
     </geometry>
```

```
    </visual>
    <collision>
     <origin xyz="0 0 0" rpy="${pi/2} 0 0"/>
     <geometry>
      <cylinder radius="${wheel_radius}" length="0.01"/>
     </geometry>
    </collision>
    <xacro:cylinder_inertia m="10" r="${wheel_radius}" h="0.005"/>
   </link>

   <joint name="${prefix}_wheel_joint" type="continuous">
    <axis xyz="0 1 0" rpy="0 0 0" />
    <parent link="base_link"/>
    <child link="${prefix}_wheel"/>
    <origin xyz="${wheel_joint} ${((base_width/2)+base_wheel_gap)*reflect} -0.005"
rpy="0 0 0"/>
   </joint>
  </xacro:macro>

  <xacro:wheel prefix="left" reflect="1" wheel_joint="0.08" />
  <xacro:wheel prefix="right" reflect="-1" wheel_joint="0.08"/>
  <xacro:wheel prefix="left_f" reflect="1" wheel_joint="-0.08" />
  <xacro:wheel prefix="right_f" reflect="-1" wheel_joint="-0.08"/>
  <joint name="base_link_joint" type="fixed">
   <origin xyz="0 0 ${wheel_radius + 0.005}" rpy="0 0 0" />
   <parent link="base_footprint"/>
   <child link="base_link" />
  </joint>

<!-- Size of square 'camera' box -->
<xacro:property name="camera_link" value="0.01" />

  <!-- Camera -->
  <link name="camera_link">
   <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
```

```
    <box size="${camera_link} ${camera_link} ${camera_link}"/>
     </geometry>
    </collision>

    <visual>
     <origin xyz="0 0 0" rpy="0 0 0"/>
     <geometry>
    <box size="${camera_link} ${camera_link} ${camera_link}"/>
     </geometry>
     <material name="red"/>
    </visual>

    <inertial>
     <mass value="0.1" />
     <origin xyz="0 0 0" rpy="0 0 0"/>
     <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
    </inertial>
  </link>


<joint name="camera_joint" type="fixed">
    <axis xyz="0 1 0" />
    <origin xyz="${base_len/2} 0 0" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="camera_link"/>
  </joint>

</robot>



<?xml version="1.0" ?>
<robot name="m2w_robot" xmlns:xacro="https://www.ros.org/wiki/xacro" >

<material name="black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
```

```
  <material name="blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="green">
    <color rgba="0.0 0.8 0.0 1.0"/>
  </material>
  <material name="grey">
    <color rgba="0.2 0.2 0.2 1.0"/>
  </material>
  <material name="orange">
    <color rgba="1.0 0.423529411765 0.0392156862745 1.0"/>
  </material>
  <material name="brown">
    <color rgba="0.870588235294 0.811764705882 0.764705882353 1.0"/>
  </material>
  <material name="red">
    <color rgba="0.80078125 0.12890625 0.1328125 1.0"/>
  </material>
  <material name="white">
    <color rgba="1.0 1.0 1.0 1.0"/>
  </material>

</robot>

<?xml version="1.0" encoding="UTF-8"?>
<launch>

  <arg name="world" default="empty"/>
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="debug" default="false"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find mybot_gazebo)/worlds/new.world"/>
    <arg name="paused" value="$(arg paused)"/>
```

```
    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
    <arg name="gui" value="$(arg gui)"/>
    <arg name="headless" value="$(arg headless)"/>
    <arg name="debug" value="$(arg debug)"/>
  </include>


  <param name="robot_description" command="$(find xacro)/xacro.py '$(find
mybot_description)/urdf/m4w_robot.xacro'"/>

    <arg name="x" default="0"/>
    <arg name="y" default="0"/>
    <arg name="z" default="0"/>
    <arg name="yaw" default="1.579232"/>


    <node name="mybot_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
          args="-urdf -param robot_description -model mybot -x $(arg x) -y $(arg y) -z $(arg
z) -Y $(arg yaw)"/>

</launch>

<?xml version="1.0" ?>
<sdf version="1.4">
  <!-- We use a custom world for the rrbot so that the camera angle is launched correctly
-->

  <world name="default">
    <!--include>
      <uri>model://willowgarage</uri>
    </include-->

    <include>
      <uri>model://ground_plane</uri>
    </include>

    <!-- Global light source -->
```

```
    <include>
      <uri>model://sun</uri>
    </include>

   <!-- Focus camera on tall pendulum -->
    <gui fullscreen='0'>
      <camera name='user_camera'>
        <pose>4.927360 -4.376610 3.740080 0.000000 0.275643 2.356190</pose>
        <view_controller>orbit</view_controller>
      </camera>
    </gui>

<model name="fire_hydrant">
    <pose>4.32 0.191 0 0 0 0</pose>
    <static>true</static>
    <link name="link">
      <collision name="collision">
        <geometry>
        <mesh>
            <uri>model://fire_hydrant/meshes/fire_hydrant.dae</uri>
          </mesh>
        </geometry>
      </collision>

      <visual name="visual">
        <geometry>
          <mesh>
            <uri>model://fire_hydrant/meshes/fire_hydrant.dae</uri>
          </mesh>
        </geometry>
      </visual>
    </link>
  </model>

<model name="person_standing">
    <pose>-0.146140 4.069 0 0 0 -0.00013</pose>
    <static>true</static>
```

```
    <link name="link">
      <collision name="collision">
        <geometry>
        <mesh>
          <uri>model://person_standing/meshes/standing.dae</uri>
        </mesh>
        </geometry>
      </collision>

      <visual name="visual">
        <geometry>
          <mesh>
            <uri>model://person_standing/meshes/standing.dae</uri>
          </mesh>
        </geometry>
      </visual>
    </link>
  </model>

<model name="stop_sign">
    <pose>3 2.954 0 0 0 -1.08</pose>
    <static>true</static>
    <link name="link">
      <collision name="collision">
        <geometry>
        <mesh>
          <uri>model://stop_sign/meshes/stop_sign.dae</uri>
        </mesh>
        </geometry>
      </collision>

      <visual name="visual">
        <geometry>
          <mesh>
            <uri>model://stop_sign/meshes/stop_sign.dae</uri>
          </mesh>
        </geometry>
```

```
      <material>
       <script>
        <uri>model://stop_sign/materials/scripts</uri>
        <uri>model://stop_sign/materials/textures</uri>
        <name>StopSign/Diffuse</name>
       </script>
      </material>
     </visual>
    </link>
  </model>
<model name="hatchback_red">
    <pose>-1.370421 -3.556199 0 0 0 -1.863615</pose>
    <static>true</static>
    <link name="link">
     <collision name="collision">
      <geometry>
      <mesh>
         <uri>model://hatchback_red/meshes/hatchback.obj</uri>
         <scale>0.01 0.01 0.01</scale>
       </mesh>
      </geometry>
     </collision>

     <visual name="visual">
      <geometry>
       <mesh>
        <uri>model://hatchback_red/meshes/hatchback.obj</uri>
       <scale>0.01 0.01 0.01</scale>
       </mesh>
      </geometry>
     </visual>
    </link>
  </model>

<model name="CONSTRUCTION BARREL">
    <pose>2.75 -4.37 0 0 0 0</pose>
    <static>true</static>
```

```
    <link name="link">
      <collision name="collision">
        <geometry>
        <mesh>
            <uri>model://construction_barrel/meshes/construction_barrel.dae</uri>
          </mesh>
        </geometry>
      </collision>

      <visual name="visual">
        <geometry>
          <mesh>
            <uri>model://construction_barrel/meshes/construction_barrel.dae</uri>
          </mesh>
        </geometry>
      </visual>
    </link>
  </model>

<model name="stop_light">

    <static>true</static>

    <!-- this pose can be overriden when including the light on another model -->
    <pose>-3.052 1.296 1 0 0 1.321406</pose>

    <link name="link">

      <collision name="collision">
        <geometry>
          <mesh>
            <uri>model://stop_light/meshes/stop_light.obj</uri>
            <scale>0.01 0.01 0.01</scale>
          </mesh>
        </geometry>
      </collision>
```

```
<visual name="frame">
  <geometry>
    <mesh>
      <uri>model://stop_light/meshes/stop_light.obj</uri>
      <scale>0.01 0.01 0.01</scale>
    </mesh>
  </geometry>
</visual>

<visual name="red">
  <pose>-0.001123 -0.082251 -0.147514 0 0 0</pose>
  <geometry>
    <sphere>
      <radius>0.1012</radius>
    </sphere>
  </geometry>
  <material>
    <script>
      <uri>model://stop_light/materials/scripts/</uri>
      <uri>model://stop_light/materials/textures/</uri>
      <name>StopLight/Light</name>
    </script>
    <ambient>1 0 0 1</ambient>
    <specular>1 0 0 1</specular>
    <!-- Turn a light on by uncommenting emissive -->
    <!--emissive>1 0 0 1</emissive-->
  </material>
</visual>

<visual name="yellow">
  <pose>-0.001123 -0.082251 -0.402 0 0 0</pose>
  <geometry>
    <sphere>
      <radius>0.1012</radius>
    </sphere>
  </geometry>
  <material>
```

```
        <script>
          <uri>model://stop_light/materials/scripts/</uri>
          <uri>model://stop_light/materials/textures/</uri>
          <name>StopLight/Light</name>
        </script>
        <ambient>1 1 0 1</ambient>
        <specular>1 1 0 1</specular>
        <!--emissive>1 1 0 1</emissive-->
      </material>
    </visual>

    <visual name="green">
      <pose>-0.001123 -0.082251 -0.655 0 0 0</pose>
      <geometry>
        <sphere>
          <radius>0.1012</radius>
        </sphere>
      </geometry>
      <material>
        <script>
          <uri>model://stop_light/materials/scripts/</uri>
          <uri>model://stop_light/materials/textures/</uri>
          <name>StopLight/Light</name>
        </script>
        <ambient>0 1 0 1</ambient>
        <specular>0 1 0 1</specular>
        <!--emissive>0 1 0 1</emissive-->
      </material>
    </visual>

  </link>
 </model>

 </world>
</sdf>

#!/usr/bin/env python3
```

```
from __future__ import print_function # package used to import print function

import roslib # contains common data structures
import sys # system module
import rospy #python client library for ROS
import cv2 # opencv module
import numpy as np # numerical array package
from std_msgs.msg import String  #representing primitive data types and other basic
message constructs
from sensor_msgs.msg import Image # #package for Camera module integrated with
robot
from cv_bridge import CvBridge, CvBridgeError # package to convert opencv image into
ros supporting image
print(sys.version)
#specify the location of the files in the code
net =
cv2.dnn.readNet("/home/vivek/project2_ws/src/mybot_motion/scripts/yolov3.weights",
"/home/vivek/project2_ws/src/mybot_motion/scripts/yolov3.cfg")
classes = []
#specify the location of the files in the code
with open("/home/vivek/project2_ws/src/mybot_motion/scripts/yolov3.txt", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))

class image_converter: # main class

  def __init__(self):  # init method
    self.image_pub = rospy.Publisher("mybot/camera/face",Image,queue_size=10)

    self.bridge = CvBridge()
    self.image_sub = rospy.Subscriber("mybot/camera/image_raw",Image,self.callback)

  #def upl(self,cv,image):
    #s3.meta.client.upload_file(cv, 'gnaneshwarb', image)
```

```
def callback(self,data): #call_back method
 try:
  img = self.bridge.imgmsg_to_cv2(data, "bgr8")
  print(img)
 except CvBridgeError as e:
  print(e)
 height, width, channels = img.shape
 blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
 net.setInput(blob)
 outs = net.forward(output_layers)
 # Showing informations on the screen
 class_ids = []
 confidences = []
 boxes = []
 for out in outs:
  for detection in out:
   scores = detection[5:]
   class_id = np.argmax(scores)
   confidence = scores[class_id]
   if confidence > 0.5:
   # Object detected
    center_x = int(detection[0] * width)
    center_y = int(detection[1] * height)
    w = int(detection[2] * width)
    h = int(detection[3] * height)

    # Rectangle coordinates
    x = int(center_x - w / 2)
    y = int(center_y - h / 2)

    boxes.append([x, y, w, h])
    confidences.append(float(confidence))
    class_ids.append(class_id)
 indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
 font = cv2.FONT_HERSHEY_DUPLEX
 for i in range(len(boxes)):
  if i in indexes:
```

```
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[i]
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label, (x, y - 5), font, 0.8, color, 2)
    cv2.imshow("Image", img)
    cv2.waitKey(1)
    try:
      self.image_pub.publish(self.bridge.cv2_to_imgmsg(img, "bgr8"))
    except CvBridgeError as e:
      print(e)

def main(args): #main function
  ic = image_converter()
  rospy.init_node('image_converter', anonymous=True)
  try:
    rospy.spin()
  except KeyboardInterrupt:
    print("Shutting down")

if __name__ == '__main__':
    main(sys.argv)

[net]
# Testing
# batch=1
# subdivisions=1
# Training
batch=64
subdivisions=16
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
```

```
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky

[convolutional]
```

```
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
```

```
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear
```

```
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
```

```
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
```

```
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
```

```
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
```

```
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky
```

```
[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
```

```
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear
```

Downsample

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

```
[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
```

```
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

######################

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
```

```
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear
```

```
[yolo]
mask = 6,7,8
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1


[route]
layers = -4

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 61



[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky
```

```
[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
```

```
pad=1
filters=512
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear


[yolo]
mask = 3,4,5
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1



[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2
```

```
[route]
layers = -1, 36



[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky
```

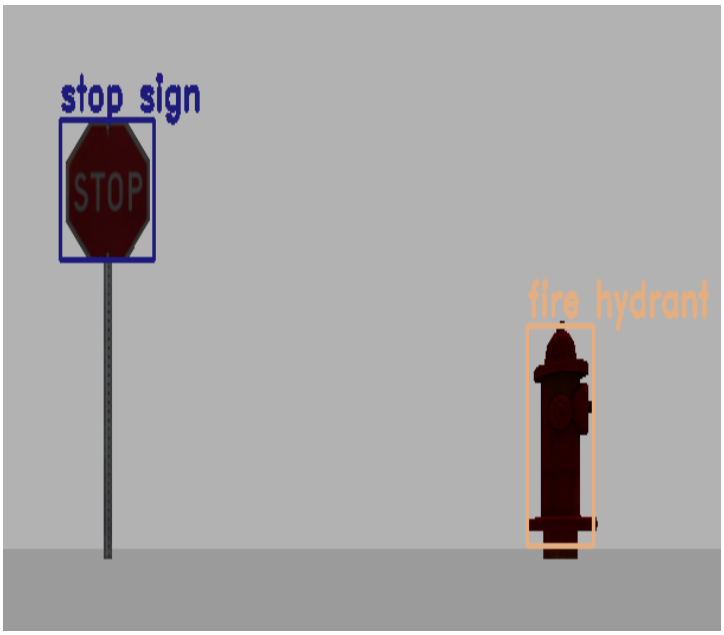
```
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear


[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

person
bicycle
```

car
motorcycle
airplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket

bottle
wine glass
cup
fork
knife
spoon
bowl
banana
apple
sandwich
orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
couch
potted plant
bed
dining table
toilet
tv
laptop
mouse
remote
keyboard
cell phone
microwave
oven
toaster
sink
refrigerator
book
clock
vase

scissors
teddy bear
hair drier
toothbrush

Robot output screenshot


stop sign
STOP
fire hydrant