ROS Perception in 5 Days

Chapter 3 Part 2: YOLO 3D Object Location

- Summary -

Estimated time to completion: **2 hours**

In this chapter, you will learn how to use YOLO in ROS to detect and position 3D objects and animals. Specifically, you will learn about:

- How to use YOLO in ROS
- How to detect people and elephants in 2D
- How to detect people and elephants in 3D

- End of Summary -

There are many deep learning libraries available. You've probably heard about: **Keras, TensorFlow, and Darknet**.

Here we will use **Darknet** because it has implemented YOLOv3, which is an object detection model. It's really fast, especially when using it in a CUDA-capable system. It's meant to do real-time detections, which for robotics is quite important.

3.4 2D YOLO Detections

We'll begin by creating our own launches that use a version of **YOLO** adapted for ROS:

▶ Execute in Shell #1

In [ ]:
```
roscd my_object_recognition_pkg
touch launch/yolo_v2_tiny.launch
```

- yolo_v2_tiny.launch -

In [ ]:
```
<?xml version="1.0" encoding="utf-8"?>

<launch>

  <!-- Use YOLOv3 -->
  <arg name="network_param_file"         default="$(find darknet_ros)/config/y
  <arg name="image" default="/camera/rgb/image_raw" />


  <!-- Include main launch file -->
  <include file="$(find darknet_ros)/launch/darknet_ros.launch">
    <arg name="network_param_file"    value="$(arg network_param_file)"/>
    <arg name="image" value="$(arg image)" />
  </include>

</launch>
```

- yolo_v2_tiny.launch -

That simple. So let's have a look at some of the parameters:

In [ ]:
```
<arg name="image" default="/camera/rgb/image_raw" />
```

Set the input **RGB ROS camera topic**. It's the most important and the only parameter you will need to change.

In [ ]:
```
<arg name="network_param_file"          default="$(find darknet_ros)/config/yo
```

There are different versions of **YOLO**: v2, v3, and v4. Each one works with different OpenCV libraries and works, in theory, better with each new version.

In this unit, we will use **v2 Tiny** (a faster version) because of the low impact on the system and fast performance.

The rest of the code is installed already in the system. If you want to access this version, you can download it from our Git: YOLO ROS TheConstruct (https://bitbucket.org/theconstructcore/yolo_tc/src/master/). If you want the original source, have a look at the following Gits:

- YOLO Darknet ROS (https://github.com/Tossy0423/yolov4-for-darknet_ros)
- Darknet ROS (https://github.com/Tossy0423/darknet_ros/tree/020b7560f9c5901605503bb59450ed7654514e9c)
- Original Source Git (https://github.com/leggedrobotics/darknet_ros)

Start the launch and see the output:

▶ Execute in WebShell #1

In [ ]:
```
# Set variables only for this course, locally you wont need it
QT_X11_NO_MITSHM=1
echo $QT_X11_NO_MITSHM
# Start yolo V2-tiny
roslaunch my_object_recognition_pkg yolo_v2_tiny.launch
```

NOTE: The first launch may result in a **GTK error**. DON'T PANIC (https://en.wikipedia.org/wiki/Don%27t_Panic:_The_Official_Hitchhiker%27s_Guide_to_the_Galaxy_Companion). Relaunch and it should work perfectly.

Move the PR2 robot around to see how it detects the different objects.

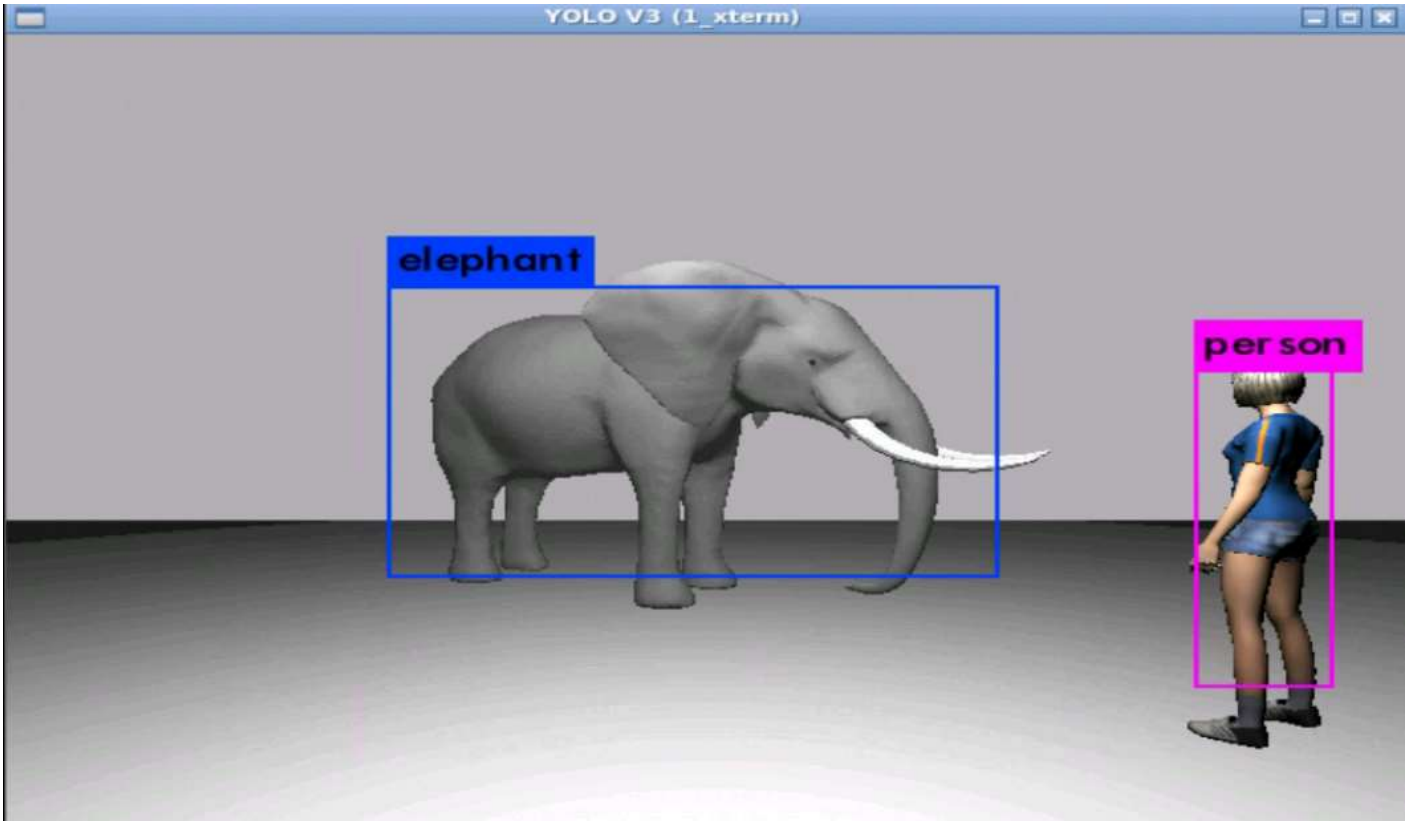
▶ Execute in WebShell #2

In [ ]:
```
roslaunch pr2_tc_teleop keyboard_teleop.launch
```

Click on the Graphical Interface icon to see the detection for the YOLO system:

You will see that in this unit, **RViz is not pre-launched**. Launch RViz in a new terminal. Note, it's not necessary in this example.

You should now see something similar to the image below:



3.5 3D Object Detection and Location with YOLO

This Darknet_ROS_3D (https://github.com/IntelligentRoboticsLabs/gb_visual_detection_3d), created by Francisco Martin (https://gsyc.urjc.es/~fmartin/), used darknet_ros in conjunction with **point-cloud data** to match the detections to 3D space and, therefore, locate more or less where the detected object is in 3D space.

This is vital for **grasping**, **autonomous navigation**, and many other applications.

The version used in this course was moded to work in **noetic**: YOLO ROS TheConstruct (https://bitbucket.org/theconstructcore/yolo_tc/src/master/).

▶ Execute in Shell #1

In [ ]:

```
roscd my_object_recognition_pkg
touch config/darknet_3d.yaml
touch launch/darknet_ros_3d.launch
```

- darknet_ros_3d.launch -

In [ ]:

```
<launch>

  <!-- Config camera image topic  -->
  <arg name="camera_rgb_topic" default="/camera/rgb/image_raw" />

  <!-- Console launch prefix -->
  <arg name="launch_prefix" default=""/>

  <!-- Config and weights folder. -->
  <arg name="yolo_weights_path"          default="$(find darknet_ros)/yolo_net
  <arg name="yolo_config_path"           default="$(find darknet_ros)/yolo_net

  <!-- ROS and network parameter files -->
  <arg name="ros_param_file"             default="$(find darknet_ros)/config/r
  <arg name="network_param_file"         default="$(find darknet_ros)/config/y

  <!-- Load parameters -->
  <rosparam command="load" ns="darknet_ros" file="$(arg network_param_file)"/>
  <rosparam command="load" file="$(find darknet_ros)/config/ros.yaml"/>
  <param name="darknet_ros/subscribers/camera_reading/topic" type="string" val

  <!-- Start darknet and ros wrapper -->
  <node pkg="darknet_ros" type="darknet_ros" name="darknet_ros" output="screer

    <param name="weights_path"          value="$(arg yolo_weights_path)" />
    <param name="config_path"           value="$(arg yolo_config_path)" />
  </node>

  <!-- Start darknet ros 3d -->
  <node pkg="darknet_ros_3d" type="darknet3d_node" name="darknet_3d" output="s
    <rosparam command="load" file="$(find my_object_recognition_pkg)/config/da
  </node>
</launch>
```

- darknet_ros_3d.launch -

- darknet_3d.yaml -

In [ ]:
```
darknet_ros_topic: /darknet_ros/bounding_boxes
output_bbx3d_topic: /darknet_ros_3d/bounding_boxes
point_cloud_topic: /camera/depth_registered/points
working_frame: camera_rgb_optical_frame
mininum_detection_thereshold: 0.3
minimum_probability: 0.3
interested_classes: ["person", "elephant", "horse", "bottle", "toothbrush", "t
```

- darknet_3d.yaml -

Comments in certain elements are important to adapt it to any robot or ROS system:

In [ ]:
```
<arg name="camera_rgb_topic" default="/camera/rgb/image_raw" />
```

We need to set the input **RGB raw camera**.

In [ ]:
```
<arg name="network_param_file"         default="$(find darknet_ros)/config/yo
```

Here, we will again set the **v2 Tiny**. It's fast, especially if you want to see point-clouds in RViz (which will have a notable effect on your system load). It's the best option.

In [ ]:
```
<rosparam command="load" file="$(find my_object_recognition_pkg)/config/darkne
```

Load the parameters for the **darknet_3d**. Set the following:

- **point_cloud_topic**: This is the ROS topic where your robot is publishing the point-cloud data. This is used to match the RGB detection in 2D and the point-cloud data in 3D to generate the bounding boxes in 3D.
- minimum_probability and mininum_detection_thereshold: We set what we consider a valid detection and generate the bounding boxes.
- interested_classes: These are the objects that you want to search for. The lower the better because of the system load impact. Here we have set **people** and **elephant.**

But how do we know which object **YOLO v2 Tiny** can detect? Check the file below:

▶ Execute in WebShell #1

In [ ]:
```
roscd darknet_ros/config
cat yolov2-tiny.yaml
```

Start the launch and see the output:

▶ Execute in WebShell #1

In [ ]:
```
# Set variables only for this course, locally you wont need it
QT_X11_NO_MITSHM=1
echo $QT_X11_NO_MITSHM
# Start yolo V3
roslaunch my_object_recognition_pkg darknet_ros_3d.launch
```

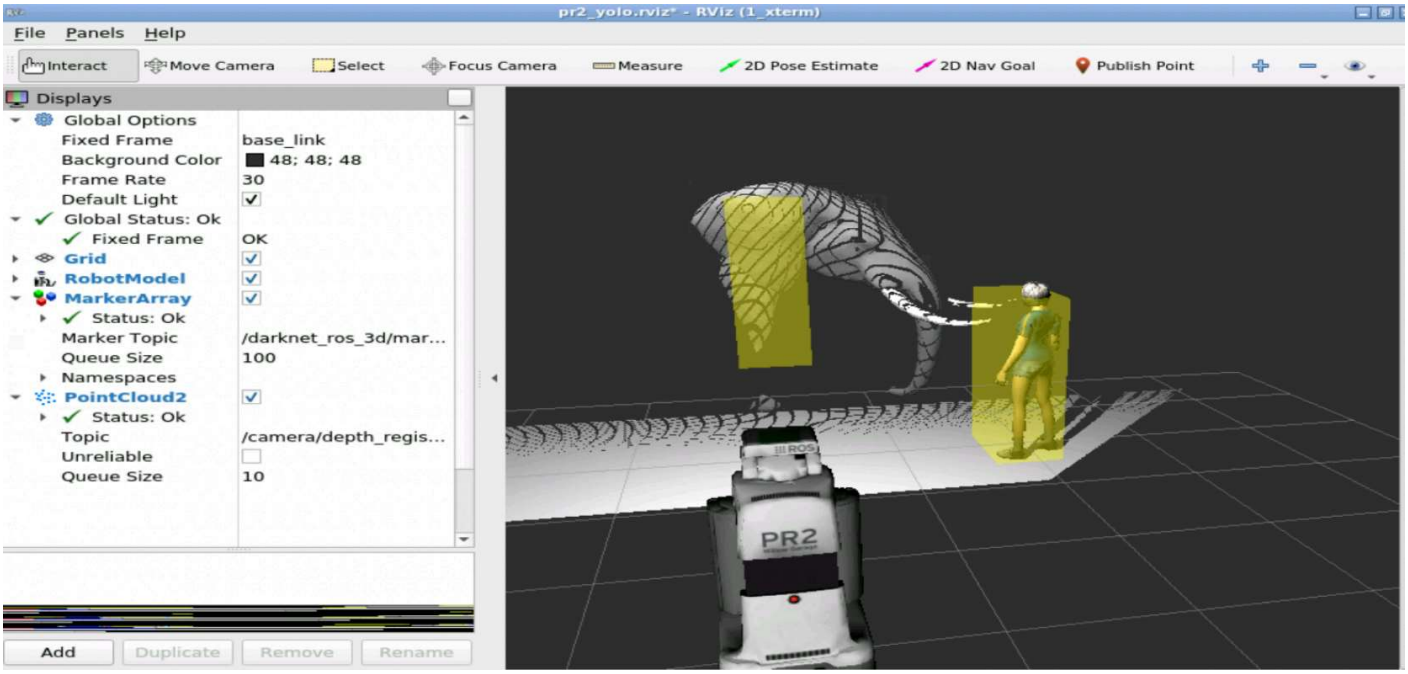
You can now open **RViz**, adding a **RobotModel**, **PointCloud2** and **MarkerArray** (for visualising the detections).

▶ Execute in WebShell #2

In [ ]:
```
rviz
```

Click on the Graphical Interface icon to open the Graphical Tools:

See below the configuration you should have:



- Exercise 3.5.1 -

- Create a Python script called **yolo_3d_data_extraction.py** that extracts the marker data generated by the darknet_ros_3d.launch and filters only the object you want. In this case, **elephant**.
- It has to be able to store multiple objects of the same type (try with **person**).
- The topic that extracts the data is **/darknet_ros_3d/bounding_boxes**.
- You have to calculate the center of the bounding box based on the max and min values.

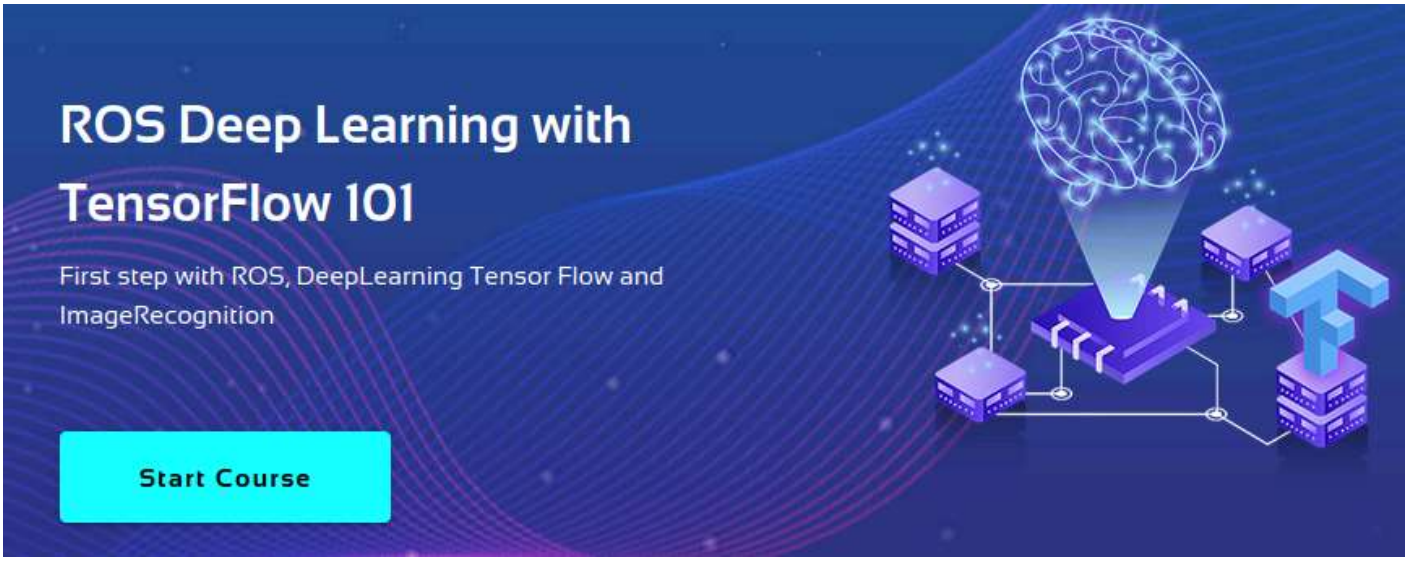
Remember to make the Python script executable in the Python script so that ROS can execute it:

▶ Execute in Shell #1

In [ ]:
```
roscd my_object_recognition_pkg
touch scripts/yolo_3d_data_extraction.py
chmod +x yolo_3d_data_extraction.py
```

- Exercise 3.5.1 -

If you want to learn more about **machine learning object detection**, check out the [TensorFlow 101 (https://app.theconstructsim.com/#/Course/28)](https://app.theconstructsim.com/#/Course/28) course:



Congratulations! You now know how to recognize objects using YOLO. In the next chapter, you will learn how to work with people-related perception.

Project

Select the project unit for this course. You can now do the fourth project exercise.

END Project