Lab04 - YOLO

MTRN4231 - UNSW School of Mechanical and Manufacturing Engineering

Preliminaries

Introduction

Task 1 - Run the camera driver

Note that this task has already been done for you but it's important to understand that a ROS2 camera package is required to subscribe to the camera feed.

```
source install/setup.bash
ros2 run v4l2_camera v4l2_camera_node
```

You should see a new topic called `/image_raw`.

Task 2 - Debugging

`rqt` can be used for debugging your ROS2 environment. Run the following command:

```
rqt
```

You can display images by going to `Plugins > Visualisation > Image View` then changing the topic to subscribe to.

Task 3 - Yolov8

Yolov8 is a CV model for object detection. A ROS2 wrapper for Yolov8 has been written and is open-source. To use it, clone the wrapper into the `src` folder of your workspace:

```
cd src
git clone https://github.com/mgonzs13/yolov8_ros.git
```

Then, install the Python dependencies **within** the cloned repository:

```
pip install -r requirements.txt
```

To run the Yolov8 node (don't forget to build and source):

```
ros2 launch yolov8_bringup yolov8.launch.py input_image_topic:='/image_raw' device:='cpu'
```

An explanation of the arguments:

- Since the `usb_cam` node publishes to `/image_raw`, redefine the topic that Yolov8 will subscribe to with the argument: `input_image_topic:='/image_raw'`.
- Since the lab computers do not have GPUs, specify: `device:='cpu'`.

Use `rqt` to display the Yolo detections on the `/yolo/detections` topic.

Task 4 - RViz

RViz can also be used to view the Yolo bounding boxes.

Run RViz:

```
ros2 run rviz2 rviz2
```

In RViz, add an image display component via the "Add" icon, and set the topic to `/yolo/dgb_image` like below:

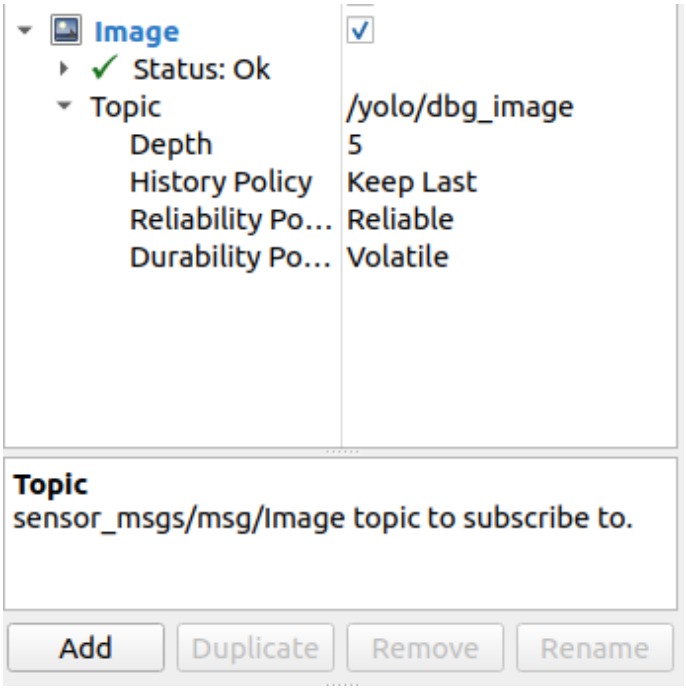

Figure 1: RViz Configuration

You may see something like the following:

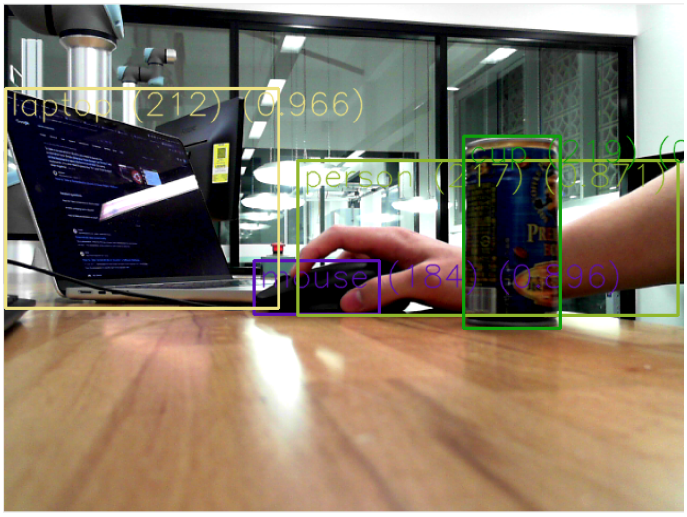

Figure 2: Yolo Output

Task 5 - Subscribing to Yolo Detections

Subscribe to Yolo detections via `/yolo/detections`. The message type is Detection.

You can also save an RViz configuration by clicking `File > Save config as`. This saves the current RViz layout and subscribed topics in a .rviz file. To launch an RViz window with the same config, run:

```
ros2 run rviz2 rviz2 -d <config>
```

```
source install/setup.bash
ros2 topic echo /yolo/detections
```

Task: Identify what each field in the message tells us about the detections, looking at the message types in the `yolov8_msgs` package. Consider what will be useful for your project.

Task 6 - Training Model

The default training model may not be robust enough for your use case of object detection.

1. Capture images from the camera using `cheese`:
   - Capture a variety of lighting conditions and angles.
   - Move the images into `my_dataset/train/images`.
2. Annotate using a labelling tool: MakeSense.ai in object detection mode. Import your images into the website and set up your labels by importing. Add more classes if it fits your project, but don't renumber existing classes if using the pre-trained model.

   When done, export annotations in YOLO format and move them to `my_dataset/train/labels`.
3. Set up YAML config file:

   ```
   path: <path_to_your_dataset>
   train: train
   val: val

   names:
   0: blue_cone
   1: large_orange_cone
   2: orange_cone
   3: yellow_cone
   4: unknown_cone
   ```
4. Train the model using CLI:

   ```
   yolo task=detect mode=train model=<source_model> imgsz=640 data=<dataset_yaml> epochs=10 batch=1
   ```

   Understand what these flags do:
   - `task`
   - `mode`
   - `model`
   - `imgsz`
   - `data`
   - `epochs`
   - `batch`
   - `name`
5. Load and test your model. The final model is located in:

   ```
   cd runs/detect/<name>/weights
   ```

   Move this model into your ROS package and configure it to use the new weights.

Resources

YOLOv8 Documentation

Task 7 - Rosbag

Rodbags allow data capture from sensors for later use. The rosbag function listens to specific topics and saves them to a '.db3' file, allowing replay on the same topic.

First, list the topics available from your nodes:

```
ros2 topic list
```

To capture a rosbag, run the following command and specify the topics to record:

```
ros2 bag record -o <rosbag_name> <topic1> <topic2> <topic...>
```

The rosbag folder can be moved to different devices or shared, and played using:

```
ros2 bag play <rosbag_name>
```

Task: Record and play a rosbag of the raw image stream. Visualize this using RViz or rqt.

Task 8 - Apply Yolo on Rosbag

We can use the recorded image stream to test image recognition without a live camera feed.

Task: Run the Yolo node to infer the rosbag data. Visualize this using RViz or rqt.

Task 9 - Launch File

Launch files run multiple nodes concurrently, define node parameters, and containerize nodes. ROS2 allows for launch files in XML and Python formats. An example is the `yolov8.launch.py` file in the `yolov8_bringup` package.

Use the following command to create a new package called `cv_bringup` and a launch directory:

```
mkdir src && cd src
ros2 pkg create --build-type ament_python cv_bringup
cd cv_bringup && mkdir launch
```

Ensure the launch files are installed with the package by adding the following line to the `data_files` list in `setup.py`:

```
from glob import glob
import os

...
# Add this to data_files (line 10)
(os.path.join('share', package_name, 'launch'), glob('launch/*.py')),
...
```

You can edit your `setup.py` and build your package.

Final Task: Write a launch file

Write a launch file to launch the Yolov8 node and the camera node together, with arguments passed to Yolov8.