YOLO Package

YOLO Package

Description:

This page discusses the *yolo_package* in detail. This package runs YOLOv3 through OpenCV's Deep Neural Networks (DNN) module and outputs YOLO detections on an input image or video.

The configuration, class, and weight files for YOLOv3 are included in this package, but they can also be downloaded from the original YOLO website. This website is also a good resource for all things YOLOv3.

Requirements:

- Ubuntu Focal Fossa
- ROS2 Foxy Fitzroy
- C++17 or higher

Configuration and Launch Files

Before using this package, make sure ALL paths in the configuration and launch files are set correctly. This will especially cause issues when cloning the repository to a new machine as the paths cloned from the remote repo are (most likely) not valid for the new local repo.

Note: All paths are set globally. Paths are also rarely set outside of configuration/launch files. This allows the user to point to different files in a more automated manner. Editing the configuration file DOES NOT require you to rebuild the package, but editing the launch file DOES!

Configuration File:
`yolo_config.yaml` is shown below:

src > yolo_package > config > ! yolo_config.yaml

```yaml
/**:
  ros__parameters:
    conf_threshold: 0.5
    nms_threshold: 0.4
    inp_width: 416 #224
    inp_height: 416 #224
    device: "cpu"
    input_type: "image"
    input_path: "/home/tahnt/T3_Repos/post_process_packages/ros2_ws/src/yolo_package/data/images/frame1130.jpg"
    write_output: true
    output_file: "/home/tahnt/T3_Repos/post_process_packages/ros2_ws/src/yolo_package/output/images/test_image.jpg"
    classes_file: "/home/tahnt/T3_Repos/post_process_packages/ros2_ws/src/yolo_package/classes/coco.names"
    model_configuration: "/home/tahnt/T3_Repos/post_process_packages/ros2_ws/src/yolo_package/cfg/yolov3.cfg"
    model_weights: "/home/tahnt/T3_Repos/post_process_packages/ros2_ws/src/yolo_package/weights/yolov3.weights"
```

ROS2 Parameters Configured:

- **conf_threshold:** The maximum allowable confidence threshold for a YOLO detection. Any detection below the set value will not be processed.
- **nms_threshold:** Non-maximum suppression threshold for the NMSBoxes function
- **inp_width:** NOT input image width! Spatial width for the convolutional neural network. Common sizes are (224×224), (227×227), (299×299), and (416x416). Lower sizes are more efficient but less accurate and larger sizes are less efficient but more accurate.
- **inp_height:** NOT input image height! Spatial height for the convolutional neural network. Common sizes are (224×224), (227×227), (299×299), and (416x416). Lower sizes are more efficient but less accurate and larger sizes are less efficient but more accurate.
- **device:** "cpu" or "gpu" - the backend device to run the CNN.
  * **Note:** Running the CNN off GPU is obviously desired for higher computational speed however this requires a specific version of OpenCV built with with a specific version of CUDA!
- **input_type:** "image" or "video" - the input type for processing
- **input_path:** Path to the input file
- **write_output:** "true" or "false" - boolean to write/save a processed output file
- **output_file:** Path to the output file
- **classes_file:** Path to the YOLO class file
- **model_configuration:** Path to the YOLO model configuration. Original YOLOv3 and YOLOv3-tiny available in package
- **model_weights:** Path to the YOLO training weights. Original YOLOv3 and YOLOv3-tiny available in package

***Launch File*:**
`yolo_launch.py` is shown below:

src > yolo_package > launch > yolo_launch.py > generate_launch_description

```python
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():

    config_path = "/home/tahnt/T3_Repos/post_process_packages/ros2_ws/src/yolo_package/config/yolo_config

    return LaunchDescription([
        Node(
            package="yolo_package",
            executable="yolo_object_detection",
            name="object_detection",
            output="screen",
            emulate_tty=True,
            parameters=[config_path]
        ),
    ])
```

Typically, the launch file does not need to be edited often. Make sure **config_path** correctly points to `yolo_config.yaml` .

YOLO Toolbox

In T3, there is also a *yolo_toolbox* repository. This is a simple centralized directory that contains the YOLOv3 model configuration, classes, and training weights. These files are required to use YOLOv3, thus packages that use YOLO will point to the files in this toolbox.

Using the Package

To run YOLO object detection follow the steps below (also included in `README.md`)

Before Use:

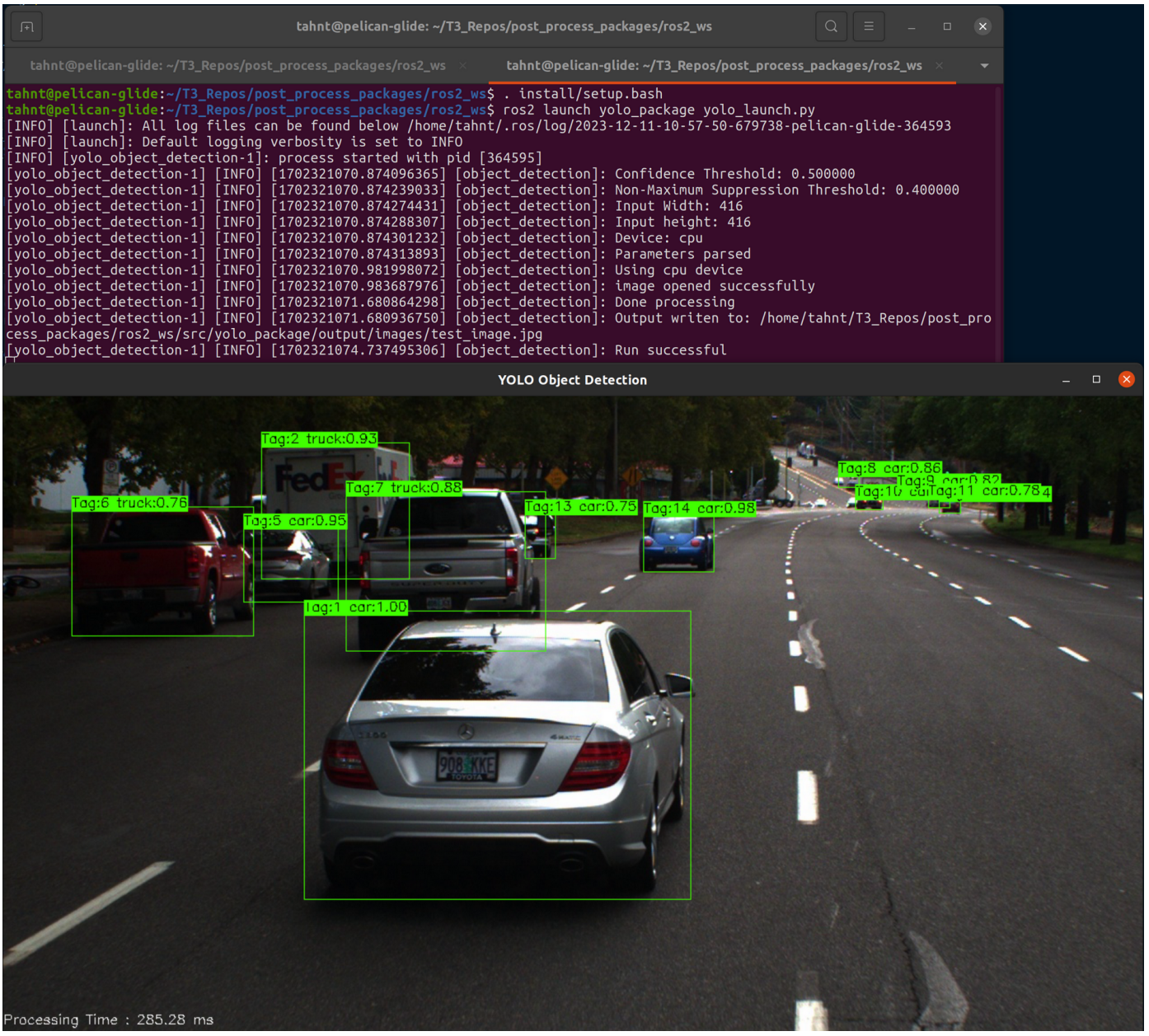
- **Make sure ALL PATHS ARE SET CORRECTLY in the launch and config files before use!**
- **These steps assume you have already created a workspace folder and a `/src` directory within it!**

Steps:

1. Navigate into the `/src` directory of your workspace and clone the repo using `git clone`
2. Navigate back into the workspace directory and source `$ source /opt/ros/foxy/setup.bash`
3. Build package `$ colcon build` or `$ colcon build --packages-select <package_name>`
4. Open a new terminal and source it `$ . install/setup.bash`
5. Run launch file `$ ros2 launch <package_name> <launch_file_name>` in this case it is `$ ros2 launch yolo_package yolo_launch.py`

If executed correctly, the output with detections will display and the terminal will look like the following (or similar):

Note: If processing a video, a window display window will also show, wait until the terminal returns 'Run successful' to process the entire video. (Will be very slow if running on CPU)



If you've set **write_output** to "true" in the config file, then an output file will be generated to the path set on **output_file**.

Note: Object labels tend to overlap making it difficult to see results. To edit what is drawn to the image (colors, labels, etc.) check out the drawPred function (shown below):

```
250  void YoloObjectDetection::drawPred(int idx, int classId, float conf, int left, int top, int right, int bottom, const Mat& frame)
251  {
252      //Load names of classes
253      vector<string> classes;
254      //string classesFile = "coco.names";
255      ifstream ifs(classesFile.c_str());
256      string line;
257      while (getline(ifs, line)) classes.push_back(line);
258
259      //Draw a rectangle displaying the bounding box
260      rectangle(frame, Point(left, top), Point(right, bottom), Scalar(0, 255, 65), 1); //Scalar(255, 178, 50)
261
262      //Get the label for the class name and its confidence
263      string label = format("%.2f", conf);
264      if (!classes.empty())
265      {
266          CV_Assert(classId < (int)classes.size());
267          label = "Tag:" + to_string(idx) + " " + classes[classId] + ":" + label; //
268      }
269
270      //Display the label at the top of the bounding box
271      int baseLine;
272      Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);
273      top = max(top, labelSize.height);
274      rectangle(frame, Point(left, top - round(1*labelSize.height)), Point(left + round(1*labelSize.width), top + baseLine), Scalar(0, 255, 65), FILLED
275      putText(frame, label, Point(left, top), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0,0,0),1.5);
276  }
```