1. Initialization

The ROS node initializes itself, sets up the node handle (nh), and parses command-line arguments to determine the image topic to subscribe to (image_topic). It also retrieves default paths for the YOLO model and class names file from ROS parameters.

2. Image Transport and Subscriber Setup

The node sets up the image transport for subscribing to the specified image topic (image_topic) and defines the subscriber (sub) to trigger the imageCallback function whenever a new image is received.

3. YOLO Model Initialization

The YOLO model file path is constructed using the ROS package path, and the YOLODetector object (detector) is instantiated with the specified model, GPU flag, and input image size.

4. Class Names Loading

The class names file path is constructed using the ROS package path, and the loadClassNames function is called to read and store class names from the file into the classNames vector.

5. Image Callback Function

The imageCallback function is invoked whenever a new image is received from the subscribed image topic (image_topic). Inside this callback, the received ROS image message is converted to an OpenCV image (cv::Mat), object detection is performed using the YOLO model, and the detected objects are annotated on the image.

In the imageCallback function:

- The ROS image message is converted to an OpenCV image using cv_bridge::toCvCopy().
- Object detection is performed using the detector.detect() method, specifying confidence and IOU thresholds.
- Detected objects are annotated on the image by drawing bounding boxes and overlaying class names using OpenCV drawing functions (cv::rectangle() and cv::putText()).
- Annotated image is converted back to a ROS image message and published on the /perception_output topic using image_pub.publish().