An Analysis of Object Detection using YOLO on The Clearpath Jackal

Mingqing Yuan
MS Robotics
Northwestern University
Evanston, IL, USA
mingqingyuan2021@u.nort
hwestern.edu

ABSTRACT

In general, autonomous driving technology can be divided into three main categories: Perception, planning, and control. As the first stage of autonomous driving, perception in the real world has always been a challenging task since it requires high accuracy and fast transmission in a constantly changing environment. Thus, Object detection has always been a key part of autonomous vehicles because it determines how the vehicle perceives the world. In this quarter, I planned to develop a mobile robotic system equipped with perception, localization, mapping, and navigation abilities using the Clearpath Jackal robot. And for the perception part, I decided to use YOLO(You Only Look Once) to accomplish the object detection task. This paper thoroughly introduces the implementation process and analysis of results for Yolo running on the Clearpath Jackal robot.

CCS CONCEPTS

• Computing methodologies→Computer graphics→ Graphics systems and interfaces • Computer systems organization→Real-time systems→Real-time operating systems

KEYWORDS

Computer vision, Neural networks, Real-time Object detection

ACM Reference format:

Mingqing Yuan. 2020. An Analysis of Object Detection using Yolo on Clearpath Jackal. *Evanston, IL, USA.*

1 Introduction

What is YOLO? In a simple word, YOLO is a single and smart convolutional neural network(CNN) for doing object detection in real-time. YOLO algorithm divides given images into regions and predicts bounding boxes as well as probabilities for each region.

Why YOLO? Just like its name, “You Only Look Once”, the algorithm requires only one forward propagation pass the neural network to make prediction and bounding box.

Since YOLO is extremely fast in real-time objection detection while reaching a high accuracy, it is wildly used for the object detection task. It is also the reason I chose to use it and tested it on the Clearpath Jackal robot.

2 Implementation

Since I used CPU instead of GPU for testing YOLO, the speed (FPS rate) should be much slower than the data shown on the website. Jackal has a slower processor than my laptop, thus I decided to test three YOLO models(YOLOv3, YOLOv4, and YOLOv4-tiny) on my laptop first and then chose which one to use on the real Jackal robot after making a comparison.

2.1 Basic Requirements

Below are the basic requirements for testing.

Hardware: Intel Realsense Camera D435i, Clearpath Jackal, Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz(laptop), Intel® Core(TM) i5-4570TE CPU @ 2.70GHz (Jackal)

Software: Python3(3.8.1), OpenCV(version==4.5.1, OpenCV’s version is important. I have not checked

other versions but YOLO did not work when I initially tested on version==4.2), Pyrealsense2, NumPy

Models: YOLOv3, YOLOv4, YOLOv4-tiny(weights and cfg files)

File: coco.names(Contains names of 80 classes)

2.2 **Set up Realsense Camera**

To get a video stream from the camera, I used pyrealsense2 libraries(SDK for Realsense camera). This SDK allows me to get color and depth image stream from the camera. For the object detection task, I need color images from the camera and then fed them into neural networks.

2.3 **Using YOLO Models in OpenCV**

OpenCV is a powerful computer vision library for handling different tasks. To use YOLO in OpenCV, the first step is to load the selected YOLO model(weights and cfg files) to the net using cv2.dnn.readNet() function. Since those models have trained with Coco dataset so they should capable of recognizing 80 different objects. Thus, the second step is to load coco.names file which contains names for those 80 objects. The third step is to use cv2.dnn.blobFromImage() function to detect objects in the image then fed them into the neural networks. The neural networks have multiple layers to extract information for every object in an image and output information such as bounding box dimension, class id(Corresponding to the name), and confidence score(Prediction). Below is an illustration of the bounding box.

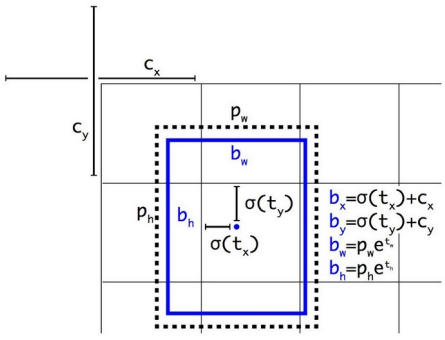


Figure 1. Bounding box

Both YOLOv3 and YOLOv4 used sigmoid function to predict the coordinates of the box's center relative to the filter. The blue box is the predicted boundary box and the black box represents the anchor. t_x,t_y, t_w, t_y are predictions from YOLO; c_x,c_y represents the top left corner of the grid cell of the anchor; p_w, p_h are width and height of the anchor; b_x,b_y,b_w,b_h are predicted boundary box; lastly, σ(t) is the box confidence score.

In a word, OpenCV provides the module to read YOLO models into neural networks then extracts information from input images and outputs bounding box, confidence score, and class_id that corresponding to the name in coco.names file.

*2.4 **Conversion to ROS Node***

Since I planned to test on the Jackal robot in the end, it is necessary to convert python code into a ROS(Robot Operating System) node so my laptop can communicate with the robot and I could see the output result in Rviz(Visualization tool for ROS). To achieve this conversion, there are three general steps. The first step is to import Image from sensor_msgs.msg(ROS message) then publish a topic with a given name in Image class. The second step is to use the cv_bridge ROS package which *converts between ROS Image messages and OpenCV images. The last step is to publish the image message then initialized the ROS node.*

2.5 **Source**

The full code and a video demo can be found on my Github page

https://github.com/dinvincible98/Jackal_SLAM

3 Analysis of Results

Since I used pre-trained YOLO models, I did not plan to evaluate the accuracy of each model. Instead, I wished to evaluate the performance(FPS rate) of those models running on my laptop and the Jackal robot since FPS is an imperative criterion for real-time object detection.

3.1 **Official Comparison Chart from COCO Website**

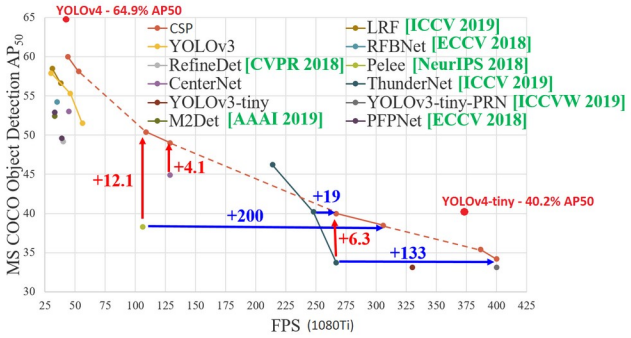


Figure 2. YOLO models comparison chart

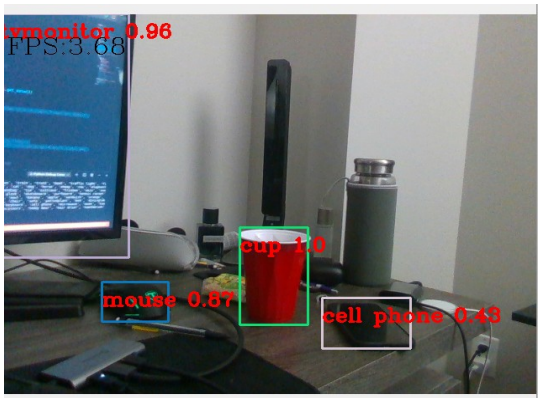
As shown in the graph, compared to YOLOv3, YOLOv4 improves the AP(Average precision) and FPS by around 10%. YOLOv4-tiny drops approximately 25% in AP compared with the other two models, but it has 10 times FPS rate than YOLOv3 and YOLOv4 when running with GTX 1080TI GPU.

3.2 **Test On Laptop**

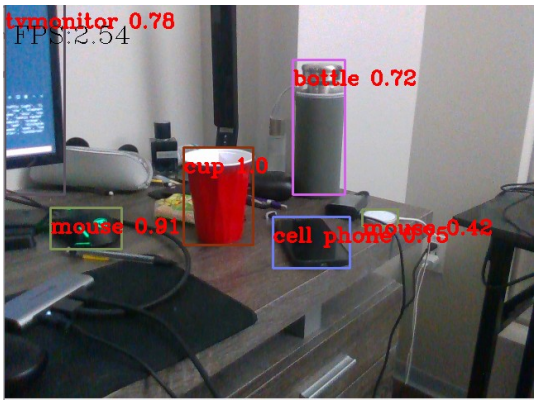
Since I did not have a powerful GPU, I need to test the performance of those three models with my laptop's CPU. Below are the results of the comparison.

Video stream output

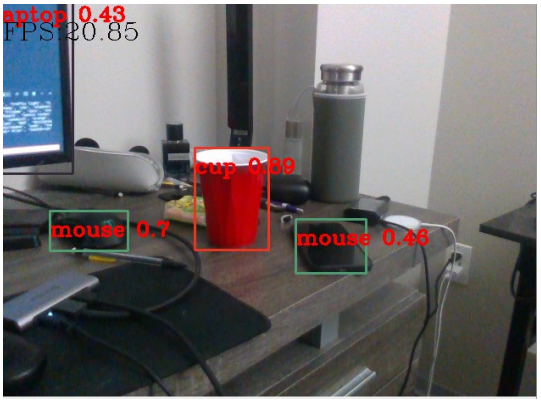
YOLOv3



YOLOv4



YOLOv4 -tiny



FPS Comparison

	FPS before converting to ROS	FPS in ROS
YOLOv3	3 - 4	Unchanged
YOLOv4	2 - 3	Unchanged
YOLOv4-tiny	20 - 21	30-31

As shown in those images, YOLOv4 has the highest accuracy than the other two models and it detects the bottle whereas the other two cannot. YOLOv4-tiny has the highest FPS but the least accuracy compared with the other two models. Since I need to run YOLO on the Jackal robot that presumably has a less powerful CPU than my laptop, I consider YOLOv4-tiny is the best choice because of its high FPS. Even though it gives a less accurate result, I

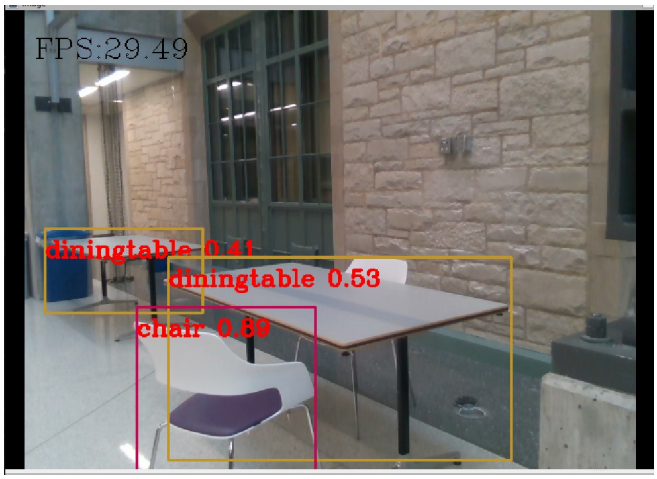
consider the trade-off here is worthy for accomplishing the real-time object detection. Besides, I surprisingly found YOLOv4-tiny could reach 30FPS in ROS whereas the other two models remained the same. Therefore, I decided to implement YOLOv4-tiny on the real Jackal robot.

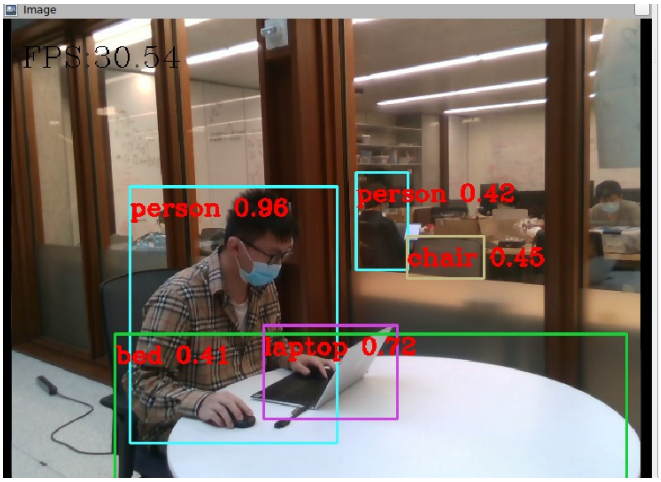
3.3 **Test On Jackal**

Before running YOLO, I need to mount the Realsense camera on Jackal.

Figure 3. Jackal with Realsense Camera on top

Below are test results in MSR Lab at Northwestern University:





As shown in the above figures, YOLOv4-tiny can reach approximately 30 FPS in ROS. Even though the precision score is not very high for some objects, this model is good enough to classify most objects. All predictions in those two images are correct except for one table which the model categorized it as a bed in the second figure.

4 **Conclusion**

Given the hardware limitation(CPU only), I thoroughly tested YOLOv3, YOLOv4 and YOLOv4-tiny based on their FPS. It turned out YOLOv4-tiny is proven to be a great model for real-time object detection model running on the Clearpath Jackal robot.

REFERENCES

[1] Peng Guei-Sian, 2019. Performance and Accuracy Analysis in Object Detection. https://scholarworks.calstate.edu/downloads/sx61dr83s

[2] Bochkovskiy Alexey, Wang Chien-Yao, Liao Hong-Yuan Mark. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. https://arxiv.org/pdf/2004.10934.pdf