DJSR Driverless Cone Detection

Introduction

In this document we discuss the processes of dataset collection, image labeling, training and testing for the perception node of our ROS package. With this, we aim to perform 3D traffic cone detection and obtain their world coordinates using just a camera.

Contents

Requirements

1. Nvidia GPU with 6GB VRAM.
2. Intel Core i5 processor.
3. 16GB RAM.
4. Ubuntu 16.04 LTS or later.
5. CUDA Toolkit and CuDNN.
6. Darknet with OpenCV and CuDNN support.

Object Detection Algorithm

We opted for an off-the-shelf algorithm, YOLOv3, which we have modified to suit our needs as it gives us reasonably good performance and accuracy on a GTX 1660 Ti with 6GB of VRAM.

With CUDA 10.2, CuDNN 7.5.6 and Nvidia driver v440.33, we get on an average 36 FPS with OpenCV enabled.

Dataset Collection

For the dataset collection, we created two tracks - one straight and another curved with a hairpin turn. The videos were then captured with a Galaxy S10 smartphone camera with f/2.4 aperture and 800 ISO at 30 FPS.

From these videos, about 450 frames were extracted at a 10 FPS delay in real-time. These 450 images were then split into sets of 350, 50 and 50 images for training, validating and testing respectively.

We performed data augmentation with image flipping and changing the values of hue, saturation, rotation, zoom and resolution. The original image resolution was kept at 1920x1080.

Image Labelling and Training

We labeled our dataset with LabelImg tool which was specifically made for YOLO. We avoided labeling far away cones as it changes the look-ahead distance of the pure pursuit controller as it may lead the car to leave the track.

To speed up training we started with the pre-trained weights for COCO dataset on which YOLOv3 was trained before. The configuration files for YOLO were also changed to output 2 classes - yellow cone and green cone. We then proceeded to train YOLO for over 2000

iterations and for 160,000 images which gives us good accuracy and precision without overfitting. The training time was about 10 hours on a Tesla K80 GPU (Google Colab).

The end result was this -

Testing

We performed testing on the test set consisting of 50 images and also on a video stream.

YOLO detects about 14 object classes per image frame which is enough for computing track center line for the pure pursuit controller. We can modify the source code of darknet to publish the coordinates of bounding boxes for a particular image to a rostopic which is subscribed by our perception node. The perception node then performs a transform that converts image coordinates to world coordinates using intrinsic camera parameters and the average depth value of an image patch.

$$y_{world} = ((y_{image} + box_{height}/2) - cy) \,/\, fy * z$$

$$x_{world} = ((x_{image} + box_{width}/2) - cx) \,/\, fx * z$$