Perception - Traffic sign detection

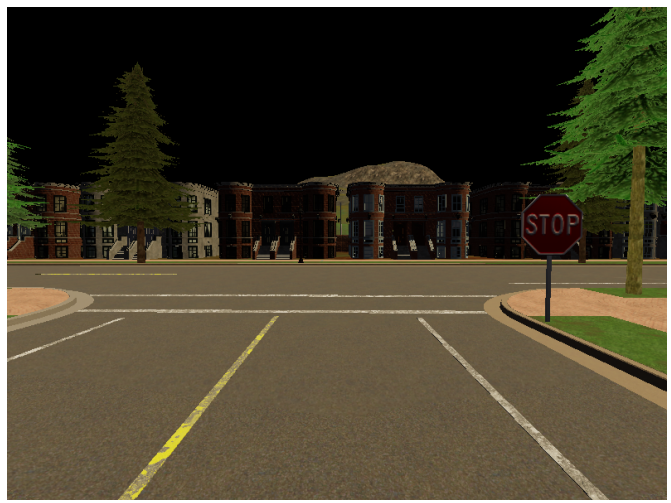
Guilherme M. Pereira
Prof. Giovanni A. Santos
Assoc. Prof. Dr.-Ing. João Paulo C. L. da Costa

Subject: Autonomous Vehicles and Artifical Intelligence
Bachelor in Software Engineeng
Professional Post-Graduate Program in Electrical Engineeng (PPEE)
Post-Graduate Program in Mechatronic Systems (PPMEC)
Universidade de Brasília (UnB)

Contents

Introduction

- Research in the field of autonomous vehicles is constantly growing. And one of the biggest causes of accidents, about 30 %, are related to speed.
- The objective of this project is to contribute to this research by integrating in a driving simulator a system capable of detecting and recognizing the speed of traffic signs on the road, making decisions that help the user to make the driving process easier and safer.
  - The identification of **traffic signs** can have a wide range of uses.
  - The simulator used in this study is CARLA and to detect traffic signals, the detection algorithm used is Yolo.



guimendesp12@gmail.com

Contents

Solutions

- Traffic Signal detection solutions
- Some examples found:
  - A Deep Neural Network to do traffic sign recognition
  - Traffic signs detection and classification in real time
  - Speed Traffic Sign Detection on the CARLA simulator using YOLO
  - Erdos-Pylot
  - Carla Simulator YOLOV3 Object Detection

Solution 1

- **A Deep Neural Network to do traffic sign recognition**
  - The approach used is deep learning.
  - The type of neural network used is a Convolutional Neural Network (CNN) paired with a linear classifier.

- Training Data needs to be Scaled and Normalized, Extended, Also Augmented, Balanced that is not possible on Carla Dataset.

Solution 2

- **Traffic signs detection and classification in real time**
  - This project is a traffic signs detection and classification system on videos using OpenCV. The detection phase uses Image Processing techniques that create contours on each video frame and find all ellipses or circles among those contours.

  - The dataset is a little bit overfitting for the classification phase and very slow. That would make it difficult to apply in the Carla simulator.

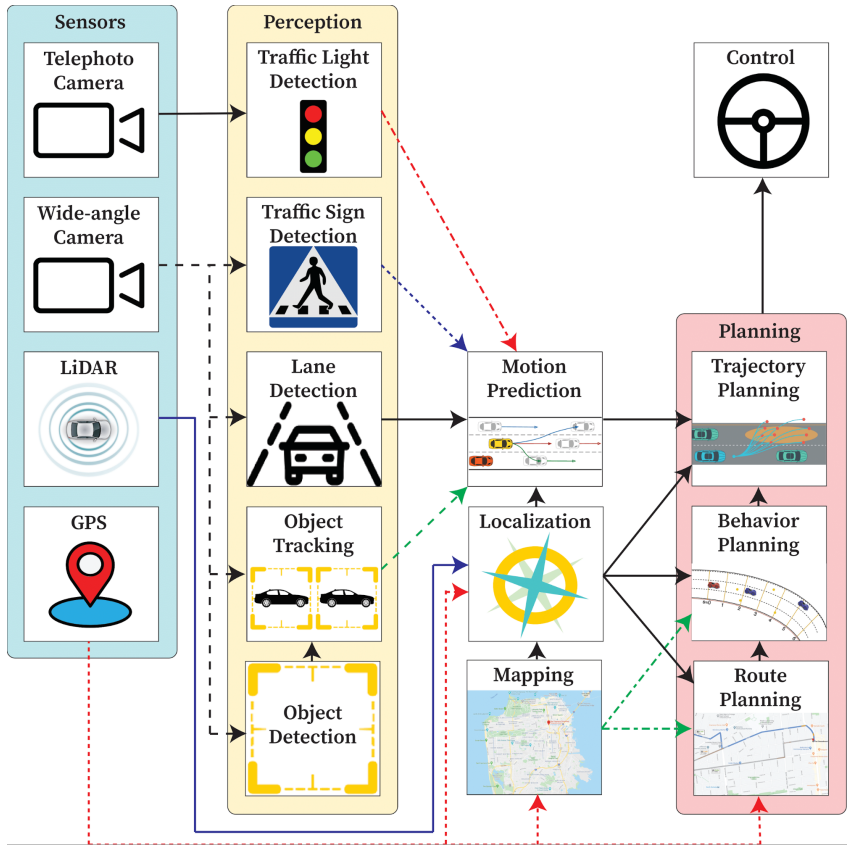
Solution 3

- **Speed Traffic Sign Detection on the CARLA simulator using YOLO**
  - The motivation that leads to this project is to build and install a system using the Darknet, Yolo model capable of detecting and recognising speed traffic signs in the CARLA simulator.
  - Carla Dataset (14 GB).

  - Code not available.
  - High computational performance for training.

Solution 4

- **Erdos-Pylot**
  - Pylot is an autonomous vehicle platform for developing and testing autonomous vehicle components (e.g., perception, prediction, planning) on the CARLA simulator and real-world cars.



  - High computational performance.

Solution 5

- **Carla Simulator YOLOV3 Object Detection**
  - Simple sample for Carla Simulator, object detection with bounding box application with yoloV3 (tensorflow 1.15.x).

  - Sometimes, it does not identify the correct objects.

Contents

CARLA

- **Carla Simulator YOLOV3 Object Detection**
  - **Requirements**
    - Carla 0.9.9
    - Python 3.7.5
    - Tensorflow-gpu 1.15.0
    - Pygame 1.9.6
    - Opencv-python 4.2.0.34
    - Opencv-python 4.2.0.34
    - Numpy 1.18.3
    - Pillow 7.1.2
  - **Project Directory Structure**

```
.CARLA_0.9.9
├── WindowsNoEditor
│   │   ├── CarlaUE4
│   │   ├── Co-Simulation
│   │   ├── Engine
│   │   ├── HDMaps
│   │   ├── PythonAPI
│   │   │   ├── carla
│   │   │   ├── util
│   │   │   ├── examples
│   │   │   │   ├── yolov3_object_detection.py
│   │   │   │   ├── tensorflow_yolov3
│   │   │   │   │   ├── carla
│   │   │   │   │   │   ├── utils.py
```

Installation and Execution

- **Setup**
  - Go to Carla PythonAPI/examples path:

```
$ git init
$ git remote add origin
    https://github.com/umtclskn/Carla_Simulator_YOLOV3_Object_Detection.git
$ git pull origin master
$ git submodule update --init --recursive
```

  - Download COCO weights from this link:

```
$ https://github.com/YunYang1994/tensorflow-yolov3/
releases/download/v1.0/yolov3_coco.tar.gz
# Extract this file under the below path:
# ..\CARLA_0.9.9\PythonAPI\examples\tensorflow-yolov3\checkpoint
```

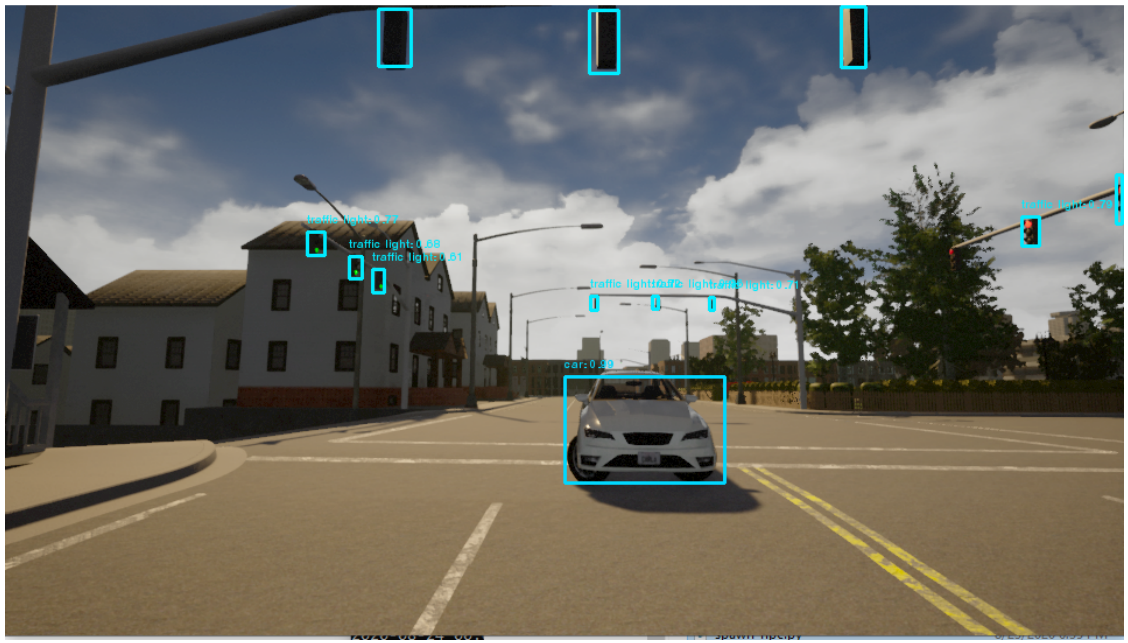
  - Run Carla Simulator YOLOV3 Object Detection:

```
# type these command at the
    ..\CARLA_0.9.9\PythonAPI\examples\tensorflow-yolov3
$ cd..
$ python convert_weight.py
$ python freeze_graph.py
# Open CarlaEU4.sh (..\CARLA_0.9.9\)
$ ./CarlaEU4.sh -windowed -ResX=800 -ResY=600 -carla-server -fps=20
    -quality-level=Low
# Run spawn actor python file for adding pedestrians or vehicles.
$ python spawn_npc.py
# Start detecting vehicles, pedestrians or bicycles.
$ python yolov3_object_detection.py
```

Contents

Running Yolov3 Detection CARLA

- The Yolo model is capable of detecting objects in real time based on a regression system, which provides classes and bounding boxes for images of traffic lights, traffic signs, cars, people, objects at speed in the execution of the algorithm.
- An RGB camera sensor was added to the CARLA vehicle obtaining in the environment information in each number of frames and leaving them ready to be evaluated by the trained model.



Implementing Yolov3 Detection Script

```
    def camera_blueprint(self):
        """
        Returns camera blueprint.
        """

        camera_bp = self.world.get_blueprint_library().find('sensor.camera.rgb')
        camera_bp.set_attribute('image_size_x', str(VIEW_WIDTH))
        camera_bp.set_attribute('image_size_y', str(VIEW_HEIGHT))
        camera_bp.set_attribute('fov', str(VIEW_FOV))

        return camera_bp

def main():
    """
    Initializes the client-side bounding box demo.
    """

    try:
        return_elements = ["input/input_data:0", "pred_sbbox/concat_2:0",
             "pred_mbbox/concat_2:0", "pred_lbbox/concat_2:0"]

# protbuf file contains the graph definition as well as the weights of the model.
        pb_file         = "tensorflow_yolov3\yolov3_coco.pb"

        # video_path      = 0
        num_classes     = 80
```

CARLA Simulator

- Demonstration

Traffic Signs detection

- Carla Lidar
- Radar Sensor
- TTC topic "time-to-collision"
- Extended Kalman Filter
- Performance of LiDAR object detection

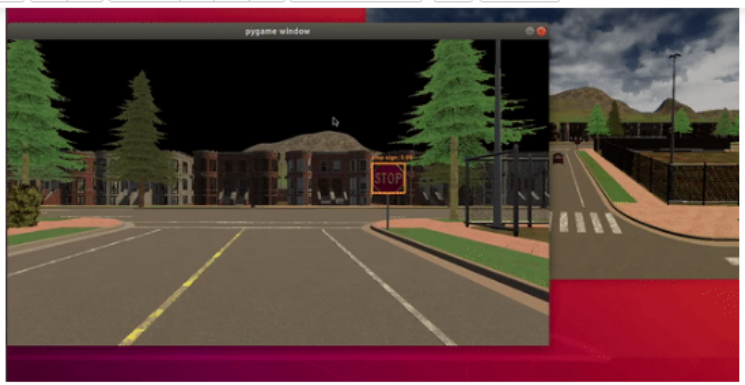
Traffic Signs distance detection

- Generation of dataset of the output of the neural network YOLO.
- Vehicle position dataset (latitude, longitude).
- Plate position on the map.

jupyter Traffic Sign Detection Analytics Carla Last Checkpoint: um dia atrás (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code Voila




```
In [1]: import numpy as np
        import pandas as pd

In [2]: df = pd.read_csv('data.csv')

In [3]: df.head()
```

Out[3]:

	accuracy	object	coordx	coordy	coordz	coordw
0	0.000000	11	590.012817	245.008469	600.899597	268.442474
1	0.635609	11	606.012817	236.008469	639.899597	271.442474
2	0.903058	11	631.614502	230.327164	674.853577	275.513306
3	0.979369	11	637.846497	228.060928	679.428589	271.221344
4	0.990305	11	642.015076	225.625229	684.618835	272.191711

Traffic Signs distance detection

- Output YOLO dataset
  - Accuracy (0.00000000 - 0.98548597)
  - Object (stop sign)
  - Coordinates bounding boxes

accuracy	object	bbox_x	bbox_y	bbox_w	**bbox_z**
0.00000000	stop sign	590.01	245.00	600.89	268.44
0.63560903	stop sign	606.01	236.00	639.89	271.44
0.90305793	stop sign	631.61	230.32	674.85	275.51
0.97936887	stop sign	637.84	228.06	679.42	271.22
0.99030471	stop sign	642.01	225.62	684.61	272.19

Traffic Signs distance detection

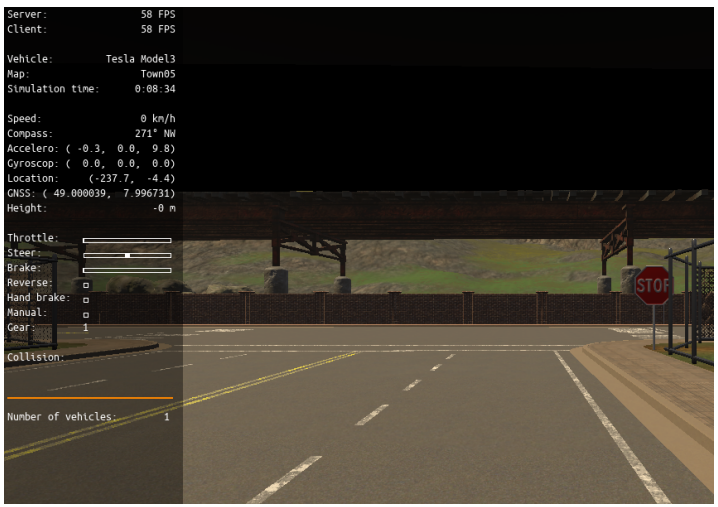
- Vehicle position and traffic sign
  - This information can be obtained using the **show_recorder_file_info.py** in simulator Carla.

Id	object	x	y	z
86	traffic.stop	-43.750	63.669	4.962
87	traffic.stop	**-257.109**	**-6.909**	15.239
88	traffic.stop	-279.638	-15.161	15.239
89	traffic.stop	142.235	6.048	16.883
90	traffic.stop	42.702	136.703	20

Id	object	x	y	z
297	vehicle.chevrolet.impala	-61.562	-0.803	30
298	vehicle.kawasaki.ninja	-162.537	-91.638	30
299	vehicle.citroen.c3	37.872	-12.850	30
300	vehicle.audi.tt	**-237.701**	**-4.430**	45
301	vehicle.harley-davidson	-141.628	84.404	30

Traffic Signs distance detection

- x (float – meters) Distance from origin to spot on X axis.
- y (float – meters) Distance from origin to spot on Y axis.
- z (float – meters) Distance from origin to spot on Z axis.
- Compare distance between coordinates



- Distance Between 2 Points

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = 19,56m$$

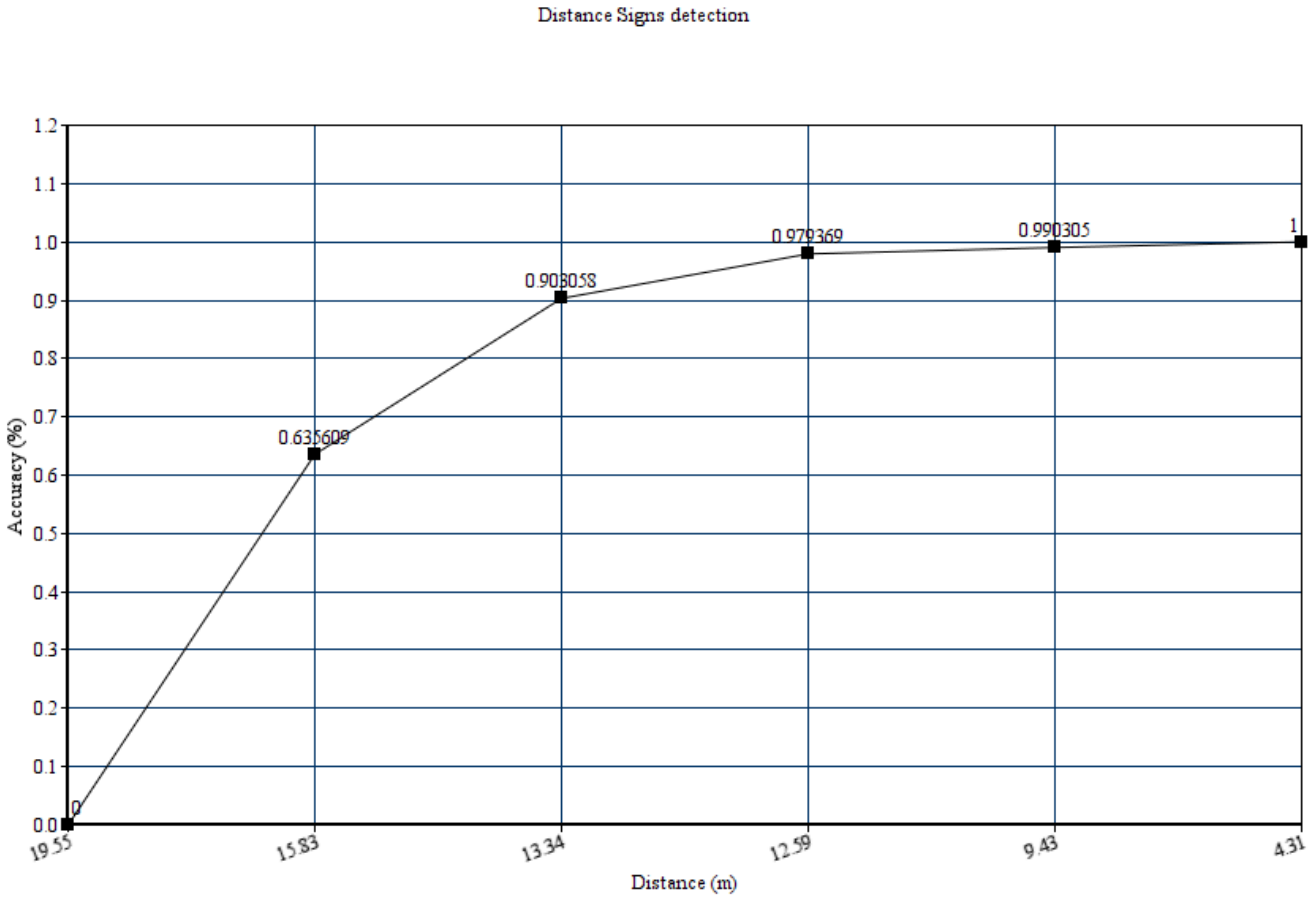
Traffic Signs distance detection

- Town05 Carla

- Videos
  - https://drive.google.com/drive/folders/13LagR8tIir67ovr2scgpEeRZDcBSwytz?usp=sharing

Traffic Signs distance detection

- Traffic.Stops
  - X - Acurracy YOLO (%)
  - Y - Object Distance (m)

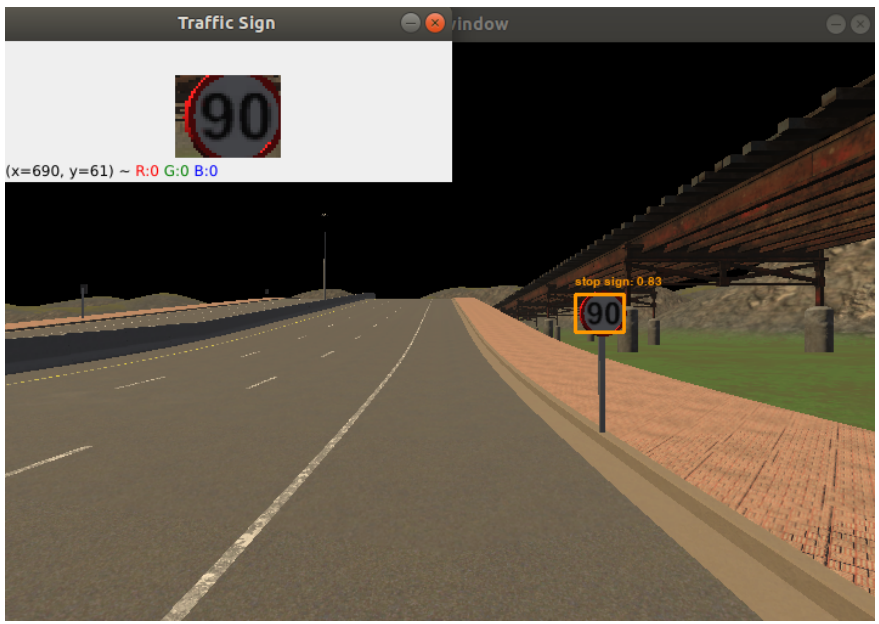


Model Training

- Image Crop detected by Yolov3 ✓
- CARLA Image collection and Dataset generation ✓
- Definition of labels for traffic signs
- Model training
- Implementation in the CARLA simulator

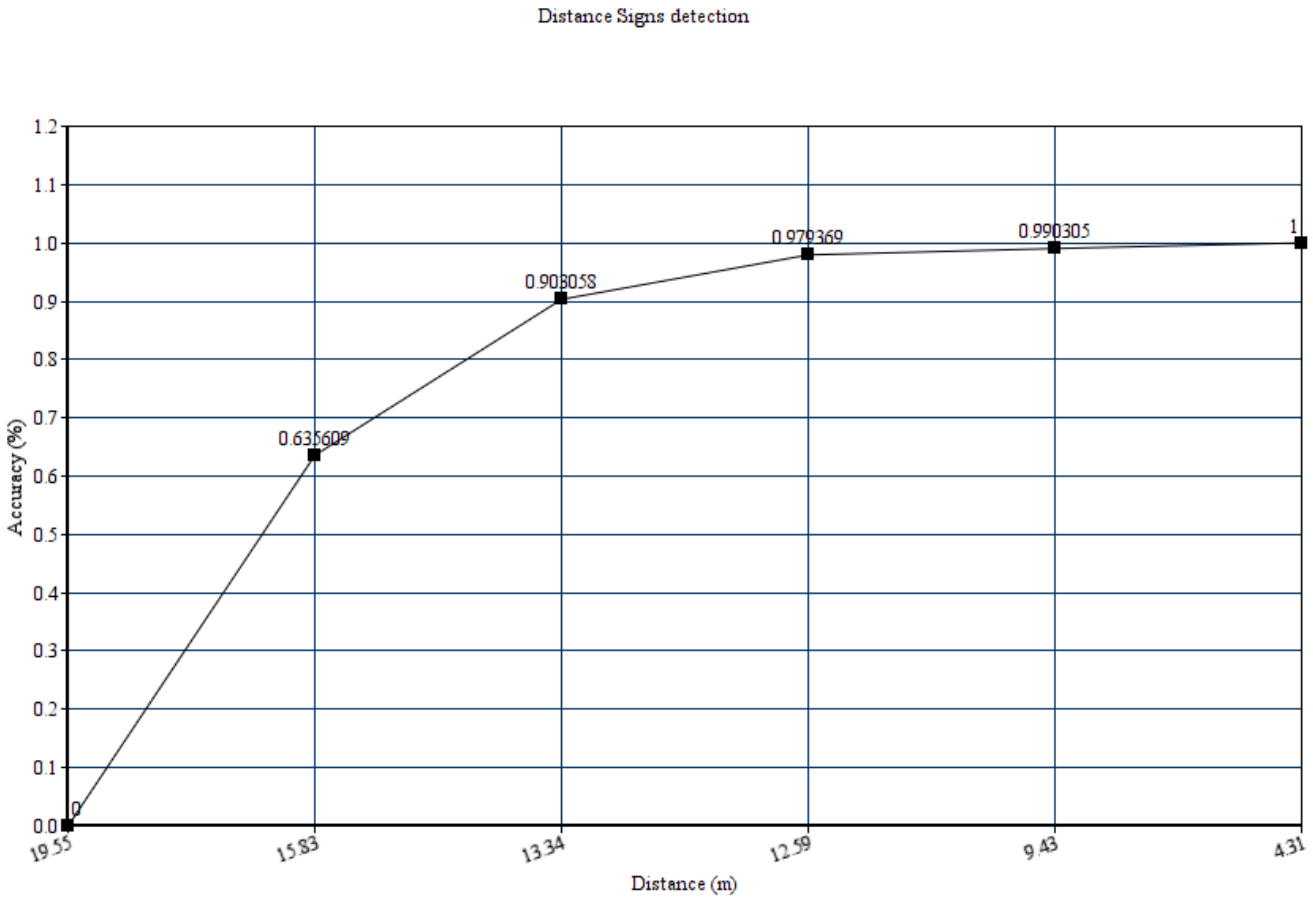
Image Crop detected by Yolov3

- Labels
  - Stop Sign
  - Speed limit (30km/h)
  - Speed limit (60km/h)
  - Speed limit (90km/h)



CARLA Dataset generation

- Traffic Signs Dataset
  - X - Acurracy YOLO (%)
  - Y - Object Distance (m)



References I

[1] *Solution 1*. https://github.com/vamsiramakrishnan/TrafficSignRecognition. Sept. 2020.

[2] *Solution 2*. https://github.com/hoanglehaithanh/Traffic-Sign-Detection. Sept. 2020.

[3] *Solution 3*. https://github.com/martisaju/CARLA-Speed-Traffic-Sign-Detection-Using-Yolo. Sept. 2020.

[4] *Solution 4*. https://github.com/erdos-project/pylot. Sept. 2020.

[5] *Solution 5*. https://github.com/umtclskn/Carla_Simulator_YOLOV3_Object_Detection. Sept. 2020.

[6] *CARLA Documentation*. https://carla.readthedocs.io/en/latest/. Sept. 2020.

[7] *YOLOV3 Tensorflow*. https://github.com/YunYang1994/tensorflow-yolov3. Sept. 2020.