33rd European Modeling & Simulation Symposium
18th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0029 ISBN 978-88-85741-57-7 © 2021 The Authors.
doi: 10.46354/i3m.2021.emss.028

Simulation of Computer Vision Based Sensor System for Autonomous Transport

Mikhail Gorobetz[1] and Ambuja Bangalore Srinivasa[2]

[1]Riga Technical University, Kaļķu iela 1, Centra rajons, Rīga, LV-1658, Latvia
[2]Riga Technical University, Kaļķu iela 1, Centra rajons, Rīga, LV-1658, Latvia

mihails.gorobecs@rtu.lv, ambuja.bangalore-srinivasa@edu.rtu.lv

Abstract

The paper provides a new insight into perception problem of autonomous transport. This study is dedicated to solve sensor fusion problem in intelligent transport system by performing tasks of object detection, real-time recognition and also determine motion parameters using solo camera as sensor. Study deals with the simulation model for a computer vision based sensor which can be very economical. A simulation model of novel system structure for recognition, speed estimation process using solo camera sensor and CNN algorithm with training is proposed in this paper. The entire computational complexity is supposed to be significantly lower in comparison with actual experimentation. From this point of view, the simulation of sensor system can be used for optimization and better results of object detection and recognition.The computer model and simulation results along with the workability of the system is presented in the paper.

Keywords: Neural Networks, Sensor simulation, Object Detection, Speed Estimation

1. Introduction

Vision systems are the heart of Advanced Driver Assistance Systems (ADAS) and various sensors act as eyes and ears for autonomous vehicles. But autonomous vehicles must be trained and tested to respond to all driving situations for which millions of scenarios must be conducted. Simulation plays a key role in such applications saving time and costs significantly. Advanced simulation tools can be used to virtually test these scenarios in less time and cost which will not only accelerate the development process but also reduce the need for physical road tests.

Artificial intelligence has become an integral part of autonomous transport system in present day. The goal of this paper is to simulate a sensor system to detect vehicles based on You Only Look Once(YOLO) Redmon et al. (2016) approach and also measure the distance and estimate speed of detected vehicle going through a defined ROI(Region Of Interest). The proposed simulated model of sensor can be used to evaluate and train the complex driving scenes. In existing systems the detection and speed estimation are being determined by different sensors for example, camera along with Convolutional Neural Network(CNN) detects object and LIDARs are used for speed estimation. The paper aims at implementing a suitable algorithm and model to achieve and optimize the discussed goal. This paper implements Convolutional Neural Network for object detection and recognition for real-time applications.

There are several recent works for object detection and recognition in real time. In the work of Oltean and Oltean (2019) real time vehicle counting using Tiny YOLO for detection and fast motion estimation for tracking by considering the region of interest. They have achieved a high accuracy of 33.5 fps in real time on

real traffic videos. In the work El Bouziady et al. (2018) propose a novel system for vehicle speed estimation from videos captured from urban traffics. Using a combination of the SIFT and KLT algorithms a precision of 0.87 and a recall of 0.92 for license plate detection is achieved. Vehicle speeds were estimated with an average error of 0.59 km/h, staying inside the +2/-3 km/h limit.

Do et al. (2021) proposes a system for vehicle speed estimation by tracking taillights of vehicles during night time using You Only Look Once as the backbone algorithm.

The work Gorobetz et al. (2018) proposes a CNN based algorithm to solve the problem of trains passing through red lights to decrease the dangerous level. The system structure is based on embedded device with artificial intelligence. A novel and effective method based on convolutional neural networks was used to achieve high-quality and real-time vehicle detection, as well as their speed using solo camera as the sensor.

2. Problem Formulation

The machine learning based applications that run an autonomous car's infotainment system receive information from the sensor data fusion systems and make predictions accordingly. Despite these advancement in the autonomous technology, as per recent news of Tesla, one of the main fundamental challenges for autonomous cars to be out on our roads are sensors and their inputs Munir et al. (2018)

That is why the main goal of the presented study is to research, develop and create a simulation algorithm to test the workability of software-intensive perception and recognition systems on the basis of intelligent embedded devices.

Novelty of this study is the development and simulation of the perception system based on solo camera, that is able not only to recognize objects, but also to measure its speed and distance without using other external sensors like LIDARs, RADARs, etc. Thus, the only input to the system are the videos fed by camera. Convolutional Neural Network (CNN) might be the best solution for object recognition. But also additional new algorithms has to be developed to analyze the position, distance and speed of the detected object.

Through this paper we try to overcome the above-mentioned challenge by developing a software and simulating the sensor based on computer vision techniques.

Following tasks are defined for this research:

1. Research of neural network learning and training algorithms for task of object detection and recognition.
2. Research of neural network learning and training algorithms for task of tracking of detected vehicles and estimate its speed.
3. Research of computer vision systems and its working methodology. Development of computer model and simulation of sensor.
4. To develop a mathematical model for object recognition and speed estimation.
5. Development of convolution neural network based algorithm for object detection and recognition.
6. Implementation of developed algorithm and testing.

3. Mathematical Model of Sensor System

In general a working system has two parts, hardware and the software. The hardware part of the simulation sensor consists of camera, processing unit (computer or embedded system), wireless transmission module to send out the data. The main goal of the simulated sensor model in this paper is to measure this data. The software part consists of Convolutional Neural Network. The functions of software part is to capture video, process frames, detect the objects and estimate speed.

The following tasks are performed for the goal achievement. First task includes the development of the structure of sensor system and development of mathematical model using Convolutional Neural Network (CNN). Second task includes the development of algorithm of CNN for detecting objects, their distance and speed. Third task is to implement the algorithm and model and analyze the results. In this paper Convolutional Neural Network based on You Only Look Once V3 algorithm is implemented for object detection and recognition. Following two steps were adopted for implementation of algorithm.

Training: Google's OpenImages V5 dataset is used for training the algorithm. This dataset has 600 different types of classes, but only few classes like cars, trucks, bicycle, person, etc were used for this implementation.

Testing: ODI V5 tooklkit is used to download necessary classes. Later on algorithm is custom trained on Keras model Chollet (2018) and tested.

3.1. Object-Oriented Mathematical Model

Developed object-oriented mathematical model of the computer-vision based sensor consists of the following components:

1. Digitalization of bitmap image

- BM = DF(CAM) = (p1 , p2, ... , pn) is a image obtained from the camera CAM and digitalized by function DF, represents as a set of bitmap pixels, where
- n = w x h, w – width of BM, px; h – height of BM, px
- each pixel $p \in BM$ contains:
  - $x_p = 1..w \in N$ X-axis coordinate in BM
  - $y_p = 1..h \in N = (0, 1, 2, 3, ..)$ Y-axis coordinate in BM
  - $c_p = CCS$-color code structure, that depends on the type of color: single value for grayscale, 3-tuple for

RGB or HSL, 4-tuple for CMYK.

2. Detection algorithm of bitmap image

- U = DET(BM) = (u1, u2, . . . , uk) – set of objects detected by the detection algorithm DET using input image BM, where
- k=0..max ∈ Z
- each $u \in U$ described by:

– $TY_u \in ty_1, \ldots, ty_m$ type of the object u, that belongs to the predefined class of objects tyi, and ty1 is a class of vehicles
– x_u = 1..$w \in N$ = (0, 1, 2, 3, ..) – localization of the detected object u in the BM by the X-axis
– y_u = 1..$h \in N$ = (0, 1, 2, 3, ..) – localization of the detected object u in the BM by the Y-axis

3. Estimation function

- $VH = EST(U| \ \forall u \in U \ \ TY_u = ty_1)$– set of parameters of the detected vehicles U calculated by the estimation function EST,
- $\forall vh \in VH, \ \ vh = < lat, \ lon, \ d, \ v, r >$, where
  lat – geographical latitude of the vehicle, deg
  lon - geographical longitude of the vehicle, deg
  d – horizontal distance from sensor to the vehicle,m
  v – speed of the vehicle, kmh
  r – course, i.e. motion direction, deg

4. Parameters for sensor

- SET = (lat0, lon0, h0, r0, CAL), - settings of the sensor, where,
  lat0 – latitude of the sensor position obtained by GNSS
  lon0 – latitude of the sensor position obtained by GNSS
  h0 – sensor position above the ground, m
  r0 – orientation (direction) of the sensor, deg
- CAL = (<x1cal, y1cal, d1cal>, . . . , <xqcal, yqcal, dqcal>) – calibration parameter
  $x^i cal$ – X coordinate of the predefined calibration point i at BM
  $y^i cal$ – Y coordinate of the predefined point i at BM
  $d^i cal$ – distance to the predefined point in meters

4. Methodology of Recognition

4.1. Simulation System Structure of Object Detection

Simulation system structure consists of(as in fig 1):

1. Capturing of object by camera
2. Sensor system simulation
3. Object detection using CNN with training
4. Distance measurement of objects with reference to frames by point object tracking
5. Speed estimation using consecutive frames
6. Transmission of output
7. Display of results with bounding boxes around detected objects and speed of vehicle

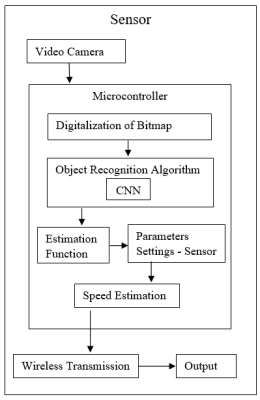

Figure 1. Simulation System structure

4.2. Proposed Algorithm

In this section architecture and algorithm with training for Convolutional neural network for object recognition block is presented. Architecture of the CNN consists of : Convolutional layer (CONV layer), Pool layer, Fully-connected layer (FC layer).

CONV layer computes the output of neurons that are connected to local regions in the input. Each neuron computing a dot product between their weights and a small region they are connected to in the input volume.

The CONV layer's parameters consist of a set of learnable filters. This layer requires four hyperparameters:
K - number of filters;
F – filter's spatial extent;
S - the stride;
P - the amount of zero padding.

Input of this layer is a 3D matrix M1 with pixel values and W1×H1×D1 size, where:
W1 – weight of the matrix before passing through the CONV layer;
H1 – high of the matrix before passing through the CONV layer;
D1 – depth of the matrix before passing through the CONV layer. For RGB picture D1=3.

After matrix M1 passed through the CONV layer the new matrix M2 with pixel values and W2×H2×D2 size is generated, where:

$$W2 = (W1F + 2P)/S + 1; \quad (1)$$

$$H2 = (H1F + 2P)/S + 1; \quad (2)$$

$$D2 = K \tag{3}$$

With parameter sharing, it introduces FFD1 weights per filter, for a total of (FFD1)K weights and K biases.

Functions of the POOL layer are: to progressively reduce the spatial size of the representation; to reduce the amount of parameters and computation in the network; and hence to also control overfitting. This layer requires two hyperparameters:
F – filter's spatial extent;
S - the stride

Input of this layer is a 3D matrix M1 with pixel values. After matrix M1 passed through the POOL layer the new matrix M2 with pixel values and W2×H2×D2 size is generated, where:

$$W2 = (W1F)/S + 1; \tag{4}$$

$$H2 = (H1F)/S + 1; \tag{5}$$

$$D2 = D1 \tag{6}$$

FC layer computes the class scores. Each neuron in this layer will be connected to all the numbers in the previous volume as in ordinary NN. This layer requires two hyperparameters: HID - Number of hidden layer neurons;
CL - Number of output classes.

Input of this layer is a 3D matrix M with pixel values and W×H×D size. Layer produces a matrix of size 1×CL where:

$$C = (c1, \ldots, c_{CL}) \tag{7}$$

$$c_1 = \{0, 1\}; \tag{8}$$

$$\sum_{i=1}^{CL} c_i = 1 \tag{9}$$

Number of weights with bias for the hidden layer: WHID=(W×H×D+1)×HID Number of weights with bias for the output layer: WCL=(HID+1)×CL

The Convolutional Neural Networks consists of the sequence of Convolutional and Pooling layers with Full-connected subnetwork at the end. General structure of the CNN: INPUT $[W0 \times H0 \times D0] \rightarrow$
$\rightarrow CONV1[K_1, F_{c1}, P_1, S_{c1}] =$
$= OUT[W_{11} = (W_0 - F + 2P)/S + 1 * H_11 =$
$= (H_0 - F + 2P)/S + 1 * H_11 * K_1 > D0, K_1/D_0 = int] \rightarrow$
$\rightarrow POOL_1[F_{p1}, S_{p1}] =$
$= OUT[W_{12} = W_{11}/F_{p1} * H_{12} = H_{11}/F_{p1} * K_1] \rightarrow$
$\rightarrow \ldots$
$\rightarrow CONVN[Kn, Fcn, Pn, Scn] = OUT[W_{n1} * H_{n1} * K_{n1}] \rightarrow$
$\rightarrow POOLN[F_{pn}, S_{pn}] =$
$= OUT[W_{n2} = W_2/F_{pn} * H_{n2} = H_2/F_{pn} * K_{n1}] \rightarrow$
$\rightarrow FC[HID, CL] = OUT[1 * CL]$

The bigger amount of layers CNN has, the more exact result can be got.

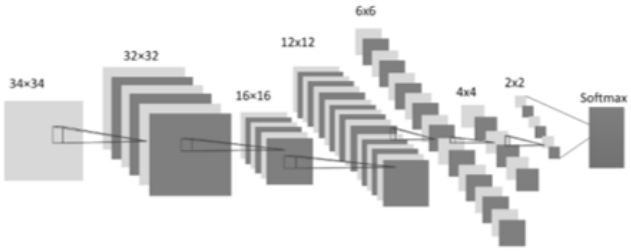

Figure 2. CNN Architecture Girshick (2015)

Once the CNN algorithm is fine tuned, the network takes as input an image (or an image pyramid, encoded as a list of images) and a list of R object proposals to score. At test-time, Real number-R is typically around 2000, although $R(\approx 45k)$ is considered. When using an image pyramid, each Region Of Interest is assigned to the scale such that the scaled Region Of Interest is closest to 2242 pixels in area.

For each test Region Of Interest r, the forward pass outputs a class posterior probability distribution p and a set of predicted bounding-box offsets relative to r (each of the K classes gets its own refined bounding-box prediction). We assign a detection confidence to r for each object class k using the estimated probability $Pr(class = k|r) \nabla = pk$. We then perform non-maximum suppression independently for each class using the algorithm and settings from R-CNN.

4.3. Speed Estimation & Distance Calculation

The following steps are adopted for speed estimation of vehicle and flowchart is as shown in fig 4

- Sensor preparation and input of video.
- Region of Interest (ROI), vehicle attributes are initialized
- Pre processing of frames takes place by convolution neural network
- ROI is extracted
- Object detection using algorithm
- Now if objects like animals or humans are detected then algorithm creates bounding boxes and displays results.
- If vehicles are detected, tracking ID is generated
- Distance calculation of the tracked vehicles takes place
- Consecutive IDs of the same vehicles are updated and difference between the points and time is calculated
- Speed is estimated using formula,

$$Speed = \frac{Distance}{Time}$$

Distance Calculation: The pre-requisite for speed calculation is distance measurement. Measuring the distance on the road with the help of installed camera and images is a whole different task. To calculate distance, angle-of-view formula is been adopted from Han et al. (2019) where a method for distance calculation is been proposed. The graphical view of the distance calculation is shown in fig 3.

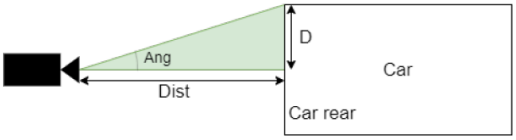

Figure 3. Graphical view of the calculation Han et al. (2019)

$$D = \frac{width}{2} \tag{10}$$

$$tan(Ang) = \frac{D}{Dist} \tag{11}$$

$$Dist = \frac{D}{tan(Ang)} \tag{12}$$

where,

- *Dist* is the distance from the camera to the car
- *Ang* is the angle-of-view, since it is the angle including the largest object whose image can fit on the frame
- *D* is the half width of the car.
- The width of the car is taken from the generated bounding box surrounding the car.

5. Implementation & Results

Expected experimental results:

1. Simulated sensor system is expected to detect and recognise pedestrians, animals, sign boards, etc on road.
2. Sensor system is expected to detect the cars and trams and their measured speed is displayed above the individual object boxes.

Object detection algorithm was applied to basic GPU

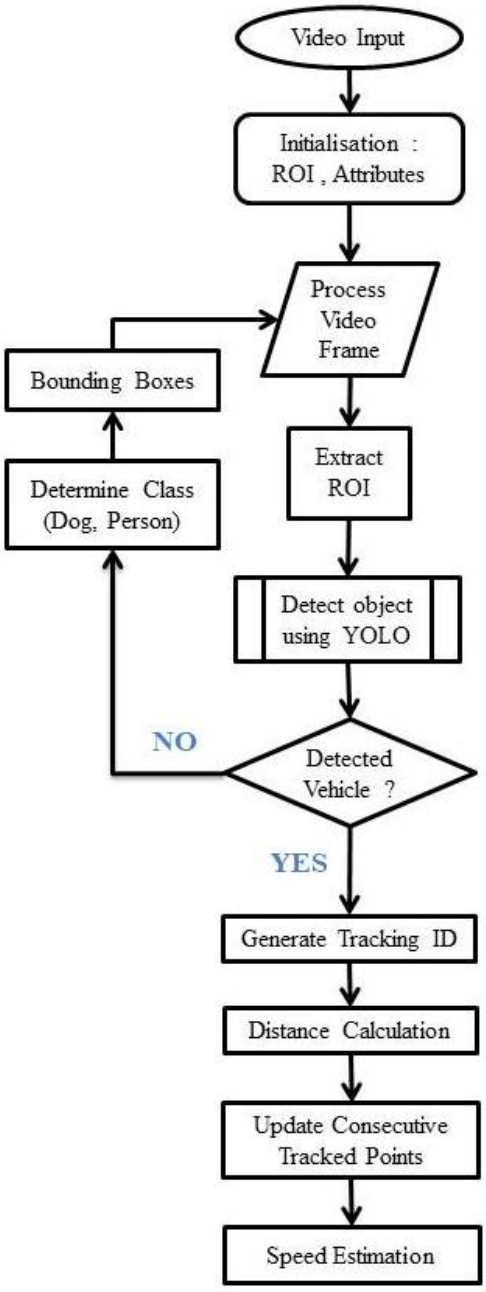

Figure 4. Flow chart of Speed Estimation

Table 1. Configuration of the computing system used

Type	Configuration
System	Lenovo G50-70
GPU	Intel® HD Graphics 4400
CPU	Intel i3-4030U
Webcam	720p or 0.3M
Operating System	Linux

configuration as shown in table 1 and satisfactory results were obtained which can be a main leverage for hardware installations. The system's software is developed using Ubuntu OS which can be a great advantage to be used in most of embedded computers.

5.1. Object Detection Results

A series of experiment was conducted to arrive at the desired results. The developed algorithm was tested for its efficiency and accuracy in both real and synthetic data. To realize the real-time detection, both webcam detection and real-time video feed experiments were conducted.

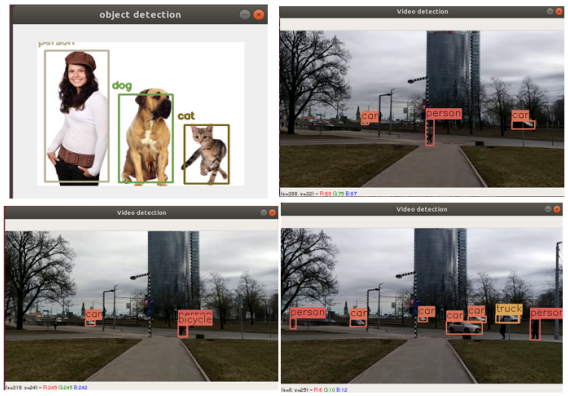

Figure 5. Object detection results

Analysis: fig 5 shows the results obtained for real-time object detection. Few sample images available on google were tested to ensure the workability of the algorithm. Algorithm was able to detect pedestrian, bicycle along with the rider, multiple cars from the video that was captured outside the university campus(Kipsala bridge).

5.2. Synthetic Data Results

To check the functionality of the algorithm in different weather conditions, fog filter and night filters were added to the video captured outside university campus and algorithm was put to test.
The fog filter was applied to the image in an increasing scale of 10% to 90% and tested.The algorithm was able to detect objects with certainty till 70% as shown in fig 6 of fog filter application and failed to detect objects after that. Similar to fog filter, one more filter to create night effects was applied. The results obtained are shown in figure 7 where a pedestrian and few vehicles near to camera are detected. the algorithm failed to detect objects after 60% of filter application.

Figure 6. Object detection with fog filter=50% and 70%

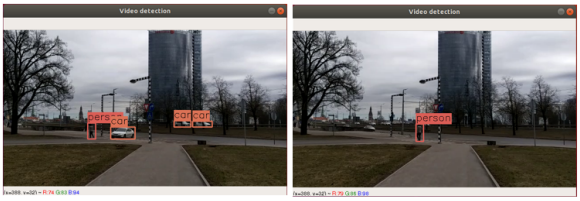

Figure 7. Object detection with night filter=50% and 60%

5.3. Speed Estimation Results

Two different experiments were conducted for obtaining speed estimation results. Tracking of objects was one of the main subtasks needed to be achieved in-order-to achieve the speed estimation. Speed estimation is calculated by using the distance of the object from the camera within a defined Region of Interest(ROI). In this paper the defined region of interest was around (3x4) meter and the defined ROI was around 2 meters away from camera.

The distance of the detected object inside the ROI was measured in every frame until the object was out of defined ROI. Later the difference in distance between the object in two consecutive frames was calculated and speed was estimated using the formula

$$Speed = \frac{Distance}{Time}$$

Figure 8. Speed estimation of car within ROI(left), outside ROI(right)

Taking practical application of the simulated sensor into consideration, a video was taken from the bridge 6-7 meters above ground, same height as where physical sensor would be installed on street lamps. Fig 8 shows results of speed estimation of car(left) which is inside defined ROI and speed estimation of cars(right) outside defined ROI

Fig 9 shows one of the interesting result obtained, it shows the speed estimation of a car which had reduced its speed near the zebra crossing and console of the tracked IDs of the vehicles which were detected during the speed estimation.

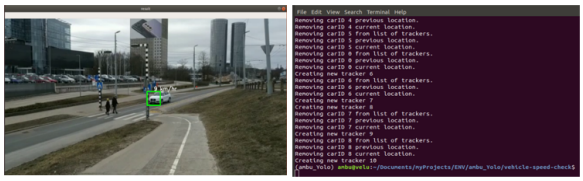

Figure 9. Speed estimation of car near zebra crossing

6. Conclusion & Future Scope

The experiment results proves that the CNN implemented and developed algorithm for computer-vision based sensor can recognize the object in real time. Experiments show that the developed model of computer vision based sensor is able to estimate speed of objects using solo camera as the sensor.

- The algorithm and model of speed estimation using only images can be used for intelligent transport system for autonomous driving.
- The algorithm and software of computer vision based sensor assures that physical sensor for speed estimation and distance calculation can be eliminated.
- The developed algorithm is able to detect objects in an image with 70% fog filter applied and 60% of night filter with certainty.
- The simulated model of object recognition with speed estimation is intended to reduce road accidents by using the sensor as a means of safety device.
- The developed sensor is intended to be used in a guidance system for disabled people, aerial surveys and in self-navigation system of UAVs and safety device for disabled.

References

Chollet, F. (2018). 16-Keras. *Deep Learning with Python*, pages 97–111.

Do, T.-h., Tran, D.-k., Hoang, D.-q., Vuong, D., and Hoang, T.-m. (2021). A Novel Algorithm for Estimating Fast-Moving Vehicle Speed in Intelligent Transport Systems. pages 499–503.

El Bouziady, A., Oulad Haj Thami, R., Ghogho, M., Bourja, O., and El Fkihi, S. (2018). Vehicle speed estimation using extracted SURF features from stereo images. *2018 International Conference on Intelligent Systems and Computer Vision, ISCV 2018*, 2018-May:1–6.

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1440–1448.

Gorobetz, M., Alps, I., Beinarovica, A., and Levchenkov, A. (2018). Algorithm of signal recognition for railway embedded control devices. *2018 IEEE 59th Annual International Scientific Conference on Power and Electrical Engineering of Riga Technical University, RTUCON 2018 - Proceedings*, pages 7–11.

Han, S., Yoo, J., and Kwon, S. (2019). Real-time vehicle-detection method in bird-view unmanned-aerial-vehicle imagery. *Sensors (Switzerland)*, 19(18):1–17.

Munir, F., Azam, S., Hussain, M. I., Sheri, A. M., and Jeon, M. (2018). Autonomous vehicle: The architecture aspect of self driving car. *ACM International Conference Proceeding Series*, (June):1–5.

Oltean, G. and Oltean, V. (2019). Towards Real Time Vehicle Counting using YOLO-Tiny and Fast Motion Estimation. (October):2019–2022.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once : Unified , Real-Time Object Detection.