Deep Learning-based Trajectory Estimation of Vehicles in Crowded and Crossroad Scenarios

Arslan Siddique[1], Ilya Afanasyev[2]
[1,2]*Innopolis University*, Kazan, Russia
[1]a.siddique@innopolis.university, [2]i.afanasyev@innopolis.ru

***Abstract*—The rapid developments in the field of Artificial Intelligence are bringing enhancements in the area of intelligent transport systems by overcoming the challenges of safety concerns. Traffic surveillance systems based on CCTV cameras can help us to achieve safe and sustainable transport systems. Trajectory estimation of vehicles is an important part of traffic surveillance systems and self-driving cars. The task is challenging due to the variations in illumination intensities, object sizes and real-time detection. We propose tracking by detection based trajectory estimation pipeline which consists of two stages: The first stage is the detection and localization of vehicles and the second stage is building associations in bounding boxes and track the associated bounding boxes. We analyze the performance of the Mask RCNN benchmark and YOLOv3 on the UA-DETRAC dataset and evaluate certain metrics like Intersection over Union, Precision-Recall curve, and Mean Average Precision. Experiments show that Mask RCNN Benchmark outperforms YOLOv3 in terms of accuracy. SORT tracker is applied on detected bounding boxes to estimate trajectories. The tracker is evaluated using mean absolute error. We demonstrate that the developed technique works successfully in crowded and crossroad scenarios.**

I. Introduction

An intelligent transport system must ensure all road users the ability to move quickly, safely and accident-free on the roads, intersections and highways. A special role is assigned to the infrastructure of the intelligent transport system, which is often equipped with a video surveillance system for traffic. Artificial Intelligence (AI) can help us in a variety of ways to solve the challenges of transport systems. Abduljabbar *et al.* [1] presented a detailed overview on the applications of AI in transportation systems. With the ever increasing trend of urbanization, traffic in cities is being multiplied and there is an urgent need for intelligent traffic surveillance systems. Hence, there is a rapid growth in the number of closed-circuit television (CCTV) cameras. They work 24 hours a day, 7 days a week and hence, generate a large amount of visual data. This data can serve as good source for traffic surveillance systems.

Trajectory estimation of vehicles is a well-known problem in the field of Artificial Intelligence and Computer Vision. Tracking by detection technique has become the preferred choice in the area of Multi-Object Tracking (MOT) [2] recently due to the developments in the field of object detection.. Although many trackers have been introduced in the modern literature [3]–[5], but the Simple Online Realtime Tracker (SORT) [6] stands out most among them because it is real time and very accurate. However, it takes bounding box detections as input which need to be estimated from a vehicle detector. The development of Convolutional Neural Networks (CNN) gave promising results for image classification. CNN based classifiers outperformed the classical machine learning and computer vision techniques. Many deep learning based object detectors have been described in literature but You Only Look once version 3 (YOLOv3) [7] and Mask region based convolutional neural network (Mask RCNN) Benchmark [8] stand out most among them because they give us good results in terms of accuracy and speed.

Analyzing the suitable datasets, we find U niversity a t Albany DEtection and TRACking (UA-DETRAC) [9] as a state of the art dataset in the area of vehicle detection and tracking. Fig. 1 shows some examples from this dataset which are clearly very complex. We are following tracking by detection pipeline in which first o f a ll, v ehicles are detected in each frame and then detected bounding boxes are tracked. Following the approach of [10], we decided to perform a thorough analysis of the performance of YOLOv3 and Mask RCNN Benchmark using several evaluation metrics like Average IoU, PR curves, Average Precision (AP) and mean Average Precision (mAP). After that we will do the trajectory estimation of vehicles in a complex and crossroad scenario from this dataset and evaluate our tracker using Mean absolute error as done by [11]. The source code is available at **https://github.com/hafizas101/Master-s-thesis**

Fig. 1. Examples from UA-DETRAC dataset presenting complex and crowded scenarios.

The survey [12] presents an overview of vision based vehicle perception systems at road intersections. However, we do not find significant work for crowded and crossroad scenarios which are very complex because of large variations in scale, lightning, occlusion and weather. We aim to address this gap in this paper and hence, consider the complex vehicle datasets. The key contributions of this paper are as follows:

1) Thorough analysis of YOLOv3 and Mask RCNN Benchmark on UA-DETRAC dataset
2) Demonstration of the success of tracking by detection pipeline for tracking vehicles in crowded and crossroad scenarios

The rest of the paper has been organized as follows: Section II describes some research work related to our project. Theoretical background and detailed methodology are explained in section III. Section IV describes implementation details and experimental setup of the project. Results have been discussed and explained in section V. Finally, we conclude in section VI.

II. Related work

Recently there have been several research works that have tried to build deep learning based vehicle detection and tracking technologies using real world data from CCTV systems. However most of the work [13]–[17] is limited to vehicle detection and localization. Some work in the field of trajectory estimation of vehicles can be found in [10], [11], [18]–[20]. Another similar work is from [21] who also used YOLOv3 with SORT tracker to track vehicles on CCTV video data stream. Sastre *et. al.* [22] also adopted tracking by detection pipeline by using modified version of Faster RCNN detector with an extended kalman filter (EKF) [23] for intelligent transportation systems. Similarly, the authors of [24] employed YOLOv3 together with kalman filter to track vehicles in the images captured by unmanned aerial vehicles (UAV).

As pointed in the first section, we are using YOLOv3 and Mask RCNN Benchmark for vehicle detection. YOLOv3 is the most recent and successful variant of a family of single stage detectors [7], [25], [26]. YOLOv3 has become a popular choice recently [13], [15], [27] because of robust performance in speed and accuracy. YOLO is an acronym for You Only Look Once. As the name suggests, this technique processes the entire image only once using a fully convolutional neural network. This network divides the image into regions and predicts bounding boxes and corresponding class probabilities based on the global context of the image. The predictions are made using a single network evaluation unlike RCNN networks [28]–[30] which require hundreds or thousands of evaluations for a single image. Among two stage detectors, Mask RCNN [31] is a very recent work which extended the Faster RCNN by adding a branch for predicting an object mask in parallel with the bounding box recognition experiencing a very small decrease in speed. Mask RCNN not only combines both target detection and segmentation in one task but also offers substantial improvement in detection accuracy as compared to other target detectors.

In the area of tracking, an Intersection over Union (IoU) tracker was proposed by E.Bochinski et. al. [32] which associated those bounding boxes as a track using greedy algorithm whose IoU value was greater than a particular threshold. They further extended IoU tracker to Visual IoU (V-IoU) tracker [33] by adding visual information which helps IoU tracker to deal with missing detections and reduces the number of ID switches and fragmentations. Then a real time tracker was introduced in [6] which is called Simple Online RealTime Tracker (SORT). It has become a very popular choice recently [11], [19] because of robust performance in speed and accuracy. It adopts frame by frame association and uses Hungarian Algorithm [34] for finding the association of vehicles in current frame with the previous frames and Kalman Filter [35] for tracking the associated vehicles. They further extended the SORT tracker to DeepSORT [36] tracker by replacing the original association metric with a metric that combines motion and appearance information from a pre-trained model. Another work is from [37] who used SORT algorithm to track objects from unmanned aerial vehicle (UAV) using computationally efficient embedding devices. A very recent work on the comparison of several tracking systems is done by [38] for vehicle re-identification.

III. Methodology

This paper aims to find s mooth t rajectory o f v ehicles in crowded and crossroad scenarios. We propose tracking by detection scheme in which first of all we will find the bounding box locations of vehicles and then track each bounding box. Fig. 2 shows the block diagram of proposed methodology.

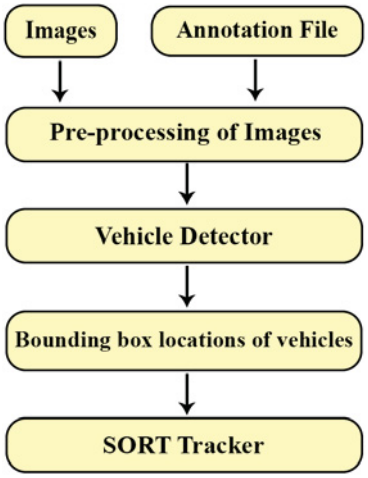

Fig. 2. Block diagram of proposed methodology

First of all, images and annotation files are passed into the pre-processing stage. The pre-processed images are passed through a vehicle detector which returns bounding box locations for cars and buses in all images. We are using YOLOv3 and Mask RCNN for vehicle detection. These bounding box locations

are fed to SORT tracker which gives us the trajectory of each vehicle. In the following sections, we describe each module in detail and provide theoretical background of our approach.

A. Mask RCNN

Mask RCNN is an instance segmentation technique that extends the features of faster RCNN. Mask RCNN adopts the same two stage strategy of Faster RCNN with the same first stage of region proposal network (RPN). However in the second stage, it outputs a binary mask as well for each RoI in parallel to bounding box calculation and class prediction. The block scheme of Mask RCNN is shown in Fig. 3.

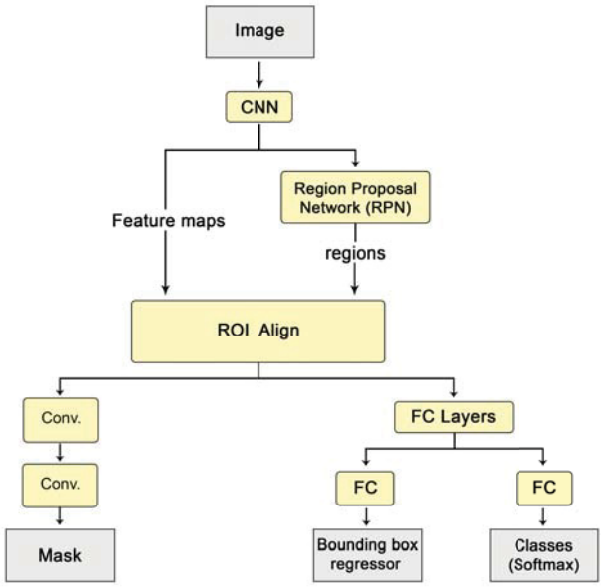

Fig. 3. Block scheme of Mask RCNN

First of all, image is passed through a pre-trained convolutional neural network (CNN) to extract image features. Then it uses another CNN called region proposal network to propose regions of interest (RoI). These RoIs together with feature maps are sent to RoI align layer so that each RoI gives fixed size feature map. The feature map is sent to two branches. One branch applies fully connected (FC) layer for object classification and detection and the second branch uses full convolutional network (FCN) to generate pixel wise mask.

B. YOLOv3

YOLOv3 is an object classifier which offers several significant differences in comparison to its predecessors. Some of these have been outlined as follows:

- YOLOv3 uses multi-label classification. So for a single region of an image, YOLOv3 can predict 'pedestrian' and 'child' which are not mutually exclusive and probabilities can sum up greater than 1. Hence, YOLOv3 replaces softmax layer with independent logistic classifiers. In this way computational complexity is reduced by avoiding the softmax function.
- YOLOv3 uses binary cross entropy loss for each label instead of mean square error. The cross entropy loss is calculated as follows:

$$-\sum_{c=1}^{M} \delta_{x \in c} log(p(x \in c)) \quad (1)$$

Where, M is the number of classes and $\delta_{x \in c}$ is logistic function that equals 1 when c is the correct class label and $log(p(x \in c))$ is the natural logarithm of probability that observation x belongs to class c.

The block diagram of YOLOv3 architecture is shown in Fig. 4. It consists of two main components:

1) Darknet-53 feature extractor
2) Feature pyramid network

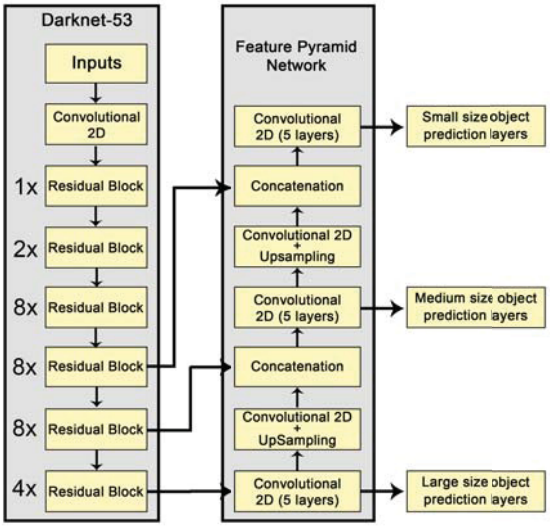

Fig. 4. Block diagram of YOLOv3 architecture

First, image is passed into a large feature extractor that consists of 53 convolutional layers architecture hence it is called Darknet-53. This architecture consists of a series of 5 residual blocks and each residual block consists of 1 x 1 and 3 x 3 convolutional filters with several skip connections. The second component is a feature pyramid network that predicts boxes at 3 different scales. The network first applies several convolutional layers at the output of feature extractor which gives us semantic information for large sized objects in the image. Then the feature map from 2 previous layers is taken and upsampled by 2 times. The upsampled features are concatenated with feature map from earlier in the network. Several convolutional layers are applied at concatenated features to give fine-grained semantic information for medium scale objects. The process is repeated to get predictions for the 3rd scale. At each scale, the detection is done by applying 1*1 detection kernel on the semantic information consisting of a 3D tensor encoding bounding box coordinates, objectness and class probabilities. Hence, the shape of the detection kernel is 1 x 1 x (3*(4+1+80)) = 1 x 1 x 255.

C. Pre-processing and Detection pipeline

Fig. 5 shows the complete illustration of Pre-processing and Detection schemes.

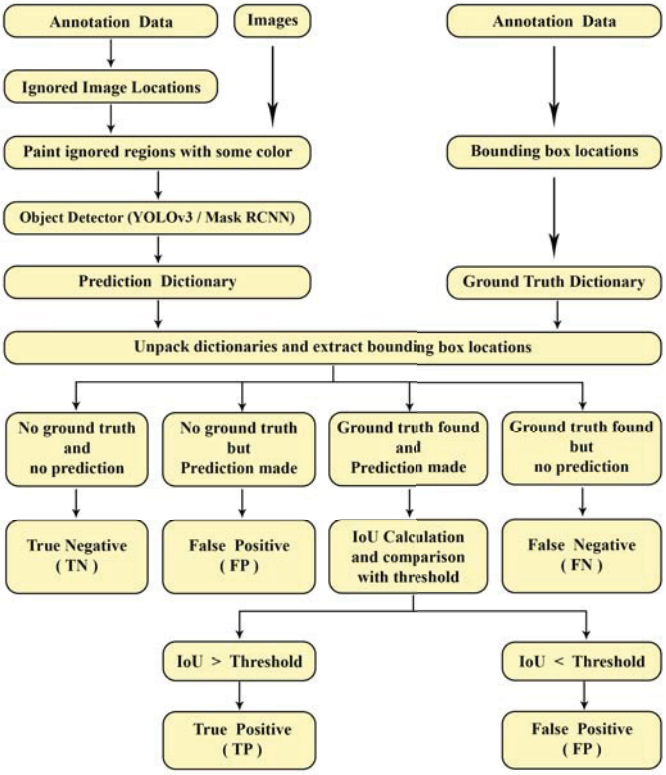

Fig. 5. Illustration of Pre-processing and Detection schemes

1) Pre-processing: This is the first stage of the whole scheme in which we assign a white pixel to all the pixels which correspond to ignored region using data from annotation file. These ignored regions represent very far away or highly occluded regions in which vehicles don't need to be detected. The Fig. 6 shows such an ignored region in white rectangular boundary in which there are large number of cars very far away and occluded. Therefore, the authors of this dataset have suggested to ignore this region.

Fig. 6. Area inside white rectangular boundary represents the region to be ignored

2) Detection: The pre-processed images are passed into an object classifier that returns only the predictions of **car** and **bus** in the image and ignores all other objects. UA-DETRAC dataset provides four labels namely car, bus, van and others. YOLOv3 and Mask RCNN both are trained on COCO dataset [39] which does not have any category of vans. So, our vehicle detectors cannot detect van. Therefore, we are considering only those predictions whose label is a car or a bus. This function returns a dictionary for each image. The dictionary consists of a list of all vehicle category labels and a list of all rectangular bounding boxes corresponding to each label. Each bounding box itself is an array of x,y locations of top left and bottom right vertices of the rectangle.

```
Image dictionary {
"category" : List of all vehicle labels in
             the image.
"locations" : List of all rectangular
              bounding boxes corresponding
              to each label.
  }
```

Then we loop through all images to create a prediction dictionary which consists of a list of Image IDs and list of corresponding image dictionaries.

```
Prediction Dictionary {
"image_id" : List of all Image IDs.
"frame_predictions" : List of all predicted
                      dictionaries.
  }
```

Then we load ground truth labels and bounding box locations from annotation data file and create a similar ground truth dictionary consisting of a list of all image IDs and a list of corresponding ground truth dictionaries.

```
Ground Dictionary {
"image_id" : List of all Image IDs.
"frame_ground" : List of all ground truth
              dictionaries.
  }
```

Then we calculate confusion matrix for both objects "car" and "bus". Assuming "car" as positive category, we check number of ground truth bounding boxes and prediction boxes for each image. Four possible situations arise:

1) If no ground truth bounding box location found and no prediction made then it is counted as a true negative (TN).
2) If ground truth bounding box is found but no prediction is made then it is counted as a False negative (FN).
3) If ground truth bounding box is not found but prediction is made then it is counted as a False positive (FP).
4) If atleast one ground truth bounding box is found and atleast one prediction is made, then for every predicted bounding box, we calculate Intersection over Union (IoU). If IoU is greater than a particular IoU threshold, then it is considered as true positive (TP) otherwise false positive (FP).

Finally, we calculate precision, recall, plot precision recall (PR) curve and calculate average precision (AP).

D. SORT tracker

In general, object tracking consists of three main steps:

1) Take initial set of object detections and create a unique ID for each of the initial detections.
2) Track each object as it moves around in frames and maintain its unique ID.
3) Create new ID for new additions and remove the IDs of disappeared objects.

Figure. 7 shows the block diagram of implemented SORT tracker together with Kalman Filter loop. Trackers represent previous frame detections whereas detections represent current frame detections. First of all, Intersection over Union matrix is calculated which represents the IoU values of bounding box of current frame (detections) with each bounding box of previous frame (trackers). Hungarian algorithm is applied on IoU matrix to find matched and unmatched trackers and detections. After obtaining the associated vehicles, Kalman Filter loop is used to estimate current position better than the detection.

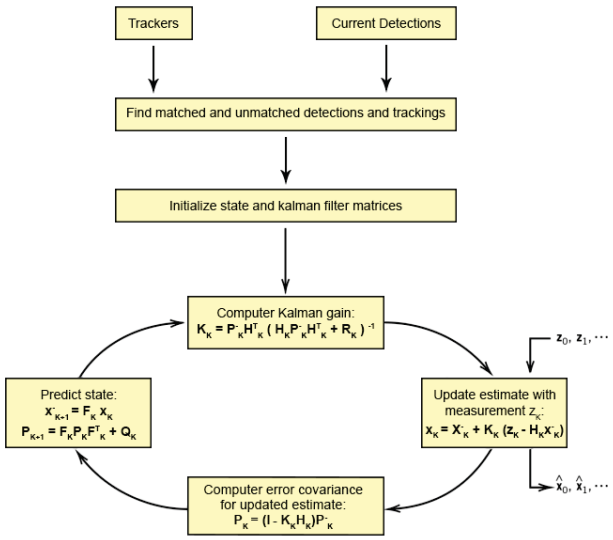

Fig. 7. SORT Tracker block diagram and Kalman Filter loop

1) In Update stage, Kalman gain is computed and our estimate is updated with measurement. Then we calculate error covariance.
2) In Prediction stage, the next state and error covariance matrices are predicted from current state, covariance matrix and state matrix.

We are using constant velocity model of Kalman Filter in which velocities are assigned to zero. The state consists of bounding box coordinates and their derivatives. This is described as:

$$X = \begin{bmatrix} x1 & \dot{x1} & y1 & \dot{y1} & x2 & \dot{x2} & y2 & \dot{y2} \end{bmatrix} \quad (2)$$

where, x1 and y1 are x and y coordinates of top left coordinate while x2 and y2 are x and y coordinates of bottom right coordinate. The state transition matrix ϕ would thus be 8 x 8 matrix and can thus be written as follows:

$$\phi = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where, dt is the time interval and is initialized as 0.2. Our measurement z_k matrix also consists of four variables because it represents bounding box location.

$$z = \begin{bmatrix} x1 & y1 & x2 & y2 \end{bmatrix} \quad (4)$$

The measurement matrix H can thus be written as follows:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The error covariance P matrix is initialized as follows:

$$P = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \quad (6)$$

Q is a noise matrix and is initialized as 8 x 8 unity matrix.

IV. Implementation and Experimental Setup

We perform various experiments on the dataset in several settings and measure some metrics like IoU, mAP and FPS. The experiments have been performed on a low-end GPU machine whose specifications are provided in Table I because of unavailability of a good GPU.

TABLE I. Specifications of Machine

System	Configuration
Operating System	Ubuntu 18.04
Processor	2.8 GHz Intel Core i7 7700 HQ
GPU	NVIDIA GTX 1050 4GB Graphics card
RAM	8 GB
Hardware memory	500 GB

A. UA-DETRAC Dataset

UA-DETRAC dataset is a challenging real-world dataset that has multiple vehicles, multiple views, weather, scale and light. It consists of 10 hours of videos captured using a Cannon EOS 550D camera at 24 different locations in Tianjin and Beijing. The videos have been recorded at 25 frames per second (FPS). Each image is 960 X 540. The dataset consists of 140131 frames in total out of which 138252 frames are labelled and have annotations. Figure. 8 shows the number of different vehicle categories of dataset in training and testing portions.

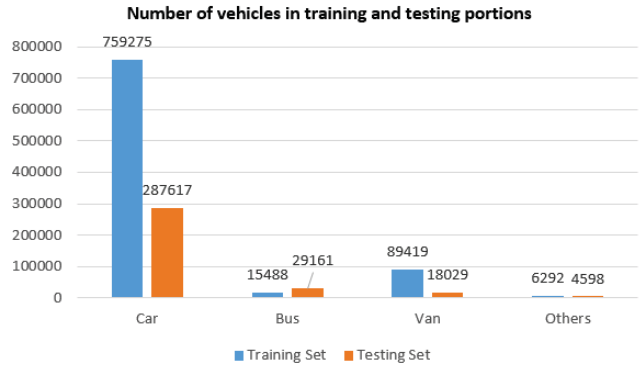

Figure 8. Number of different vehicle categories in training and testing portion

It is clear that majority of the vehicles are cars. For the simplicity of algorithm and experiment, we only consider two categories namely ”Car” and ”Bus”. All the bounding boxes from ground truth data which have ”van” or ”others” as their label, they are ignored. There are 60 sequences in training set and 40 sequences in testing set.

While working with UA-DETRAC dataset, we found missing annotations for several images. In the testing portion, we found missing annotations only for sunny and night data images. The sequence number and frame number details of these images for test portion have been provided in Table II. We deleted these images for our analysis since their annotations are not available. In the training portion, we found missing annotations for night, cloudy and rainy data images. The sequence number and frame number details of these images for training portion have been provided in Table III. No missing annotation has been observed in Sunny data images in training portion.

B. YOLOv3 Implementation

We decided to run YOLOv3 using OpenCV deep neural networks (DNN) module because it gives much more flexibility and control on the information returned by YOLOv3 algorithm like bounding box locations, confidence scores etc. We utilized pretrained YOLOv3 weights and configuration file into OpenCV Deep neural network (DNN) module. However in order to use this method, the version of OpenCV must be greater than 3.4.2 because the older versions of OpenCV cannot load YOLO into their DNN module. Fig. 9 shows qualitative results of applying YOLOv3 on a crossroad image.

TABLE II. DISTRIBUTION OF IMAGES WHOSE ANNOTATIONS ARE MISSING IN TRAINING PORTION

Weather	Sequence ID	Frame numbers
Cloudy	MVI_39931	677-835, 977-1005
	MVI_40152	1607-1610
	MVI_40162	322-360
	MVI_40211	222-235, 427-430, 447-455
	MVI_40213	1167-1170, 1287-1290
Night	MVI_39761	542-610, 897-1080, 1297-1380
	MVI_39781	337-340
	MVI_39811	207-260, 342-510, 597-655, 717-720, 787-1070
	MVI_40851	162-295
	MVI_40991	272-335, 1567-1655
	MVI_40992	487-490, 1577-1590
Rainy	MVI_63544	892-1095

TABLE III. DISTRIBUTION OF IMAGES WHOSE ANNOTATIONS ARE MISSING IN TESTING PORTION

Weather	Sequence ID	Frame numbers
Night	MVI_40743	1630
	MVI_40763	1742-1745
	MVI_40793	1960
Sunny	MVI_39051	992-1060
	MVI_39211	477-570
	MVI_39271	77-80

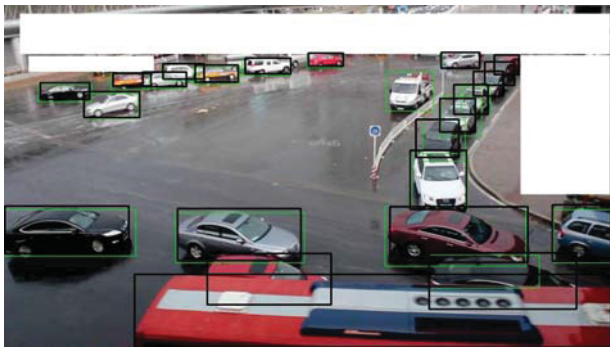

Fig. 9. Qualitative result of applying YOLOv3 on a pre-processed image. Black boxes correspond to ground truth objects and green colored boxes correspond to predicted objects. However, white boxes represent ignored region in the image.

C. Mask RCNN Implementation

We use Mask RCNN benchmark [8] which is actually the implementation of Mask RCNN [31] object detection and segmentation algorithms in PyTorch 1.0. PyTorch is a very low level API as compared to Tensorflow and Keras and is focused on direct work with array expressions. Hence, it is much faster than Keras and Tensorflow. This makes Mask RCNN benchmark several times faster than the original implementation of Mask RCNN for object detection and segmentation. The algorithm generates bounding boxes and segmentation masks for each instance of an object in the image. Figure. 10 shows the qualitative result of applying Mask RCNN Benchmark technique on a crossroad image.

D. SORT Tracker Implementation

SORT tracker consists of two main stages: The first stage is data association which uses Hungarian algorithm to associate

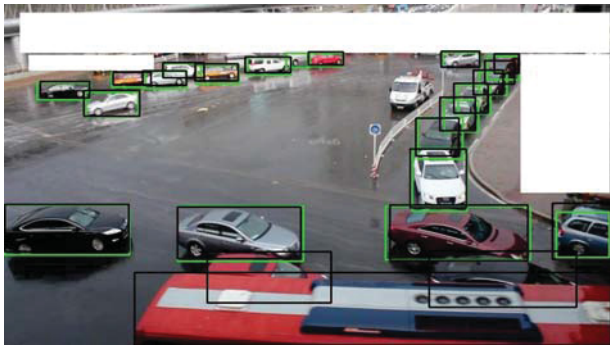

Fig. 10. Qualitative result of applying Mask RCNN Benchmark on a pre-processed image. Black boxes correspond to ground truth objects and green colored boxes correspond to predicted objects. However, white boxes represent ignored region in the image.

the vehicles in current frame to the vehicles in previous frame. The second stage is standard Kalman filter with constant velocity motion and linear observation model where we take the bounding coordinates (u, v, w, h) as direct observations of the object state. An efficient implementation of SORT tracker based vehicle tracking is provided by [40]. The implemented SORT tracker can be described in pseudo-code of the Algorithm 1.

Algorithm 1: SORT Tracker

Input: Detections as list of bounding box locations in current frame
Output: Centroids of tracked bounding boxes.

```
if This is first frame then
    Initialize an empty list of trackers
else
    Extract the detections of previous frame as trackers
Find Intersection over Union between each tracking
  and detection forming an IoU matrix
Use linear_assignment function from sklearn module
  to find matched vehicle indices using Hungarian
  algorithm
if IoU for matched vehicles < 0.3 then
    Assign them as unmatched trackers and detections.
else
    Assign them as matched vehicles.
Initialize state and Kalman filter matrices to start a
  Kalman filter loop for a vehicle.
Predict next state and update your state using current
  detection as measurement.
if Consecutive matches > min_hits and Consecutive
  unmatched detections < maxDisappeared then
    Compute and append centroid of this detection
```

Experiments for tracking have been performed on first 120 frames of sequence MVI_40852 from Cloudy portion of test data. The reason to choose this sequence is that it represents a crowded and crossroad scenario. We only need a small number of frames to demonstrate our trajectory algorithm. Hence we decided to work with first 120 frames in order to demonstrate tracking results. The resultant trajectory obtained from SORT tracker can be seen in Fig. 11 and video demo is available at https://youtu.be/HEjMSuSSheA

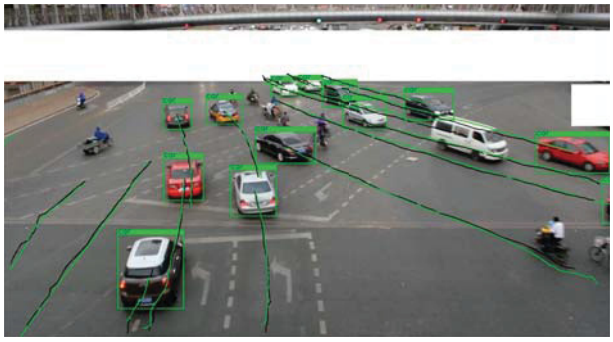

Fig. 11. Comparison of ground truth trajectory with trajectory based on SORT tracker. Black color path represents ground truth trajectory whereas green color path represents the generated trajectory from SORT tracker. White color rectangles represent the region to be ignored.

V. Results and Discussions

Fig. 9 and Fig. 10 show the qualitative results of YOLOv3 and Mask RCNN algorithms on a challenging real scenario crossroad image from rainy portion of UA DETRAC test dataset. Black boxes indicate ground truth objects and in total there are 22 labelled cars, 1 bus and 1 van according to ground truth annotation file. Since we are not considering vans in our experiments, hence we do not mark it as ground truth. YOLOv3 detects 19 cars in total one of them is actually a van which is considered as car by YOLOv3 algorithm. So, YOLOv3 fails to detect the bus and 4 cars. However, Mask RCNN is able to detect 20 cars and special thing is that it does not categorize van as car or bus. Bus is missed by both algorithms because it does not have a clear picture from the front. Both algorithms were trained on COCO dataset which has much higher number of front and back pictures of vehicles as compared to side pose. Also, it is noted that Mask RCNN is much better at recognizing small objects than YOLOv3 because Mask RCNN is able to detect all far away cars but misses the front cars hidden behind a bus. Fig. 12 and 13 give full statistics about the number of cars and buses respectively in ground truth and predictions made by both algorithms.

The number of cars detected by Mask RCNN is even greater than number of cars in ground truth for sunny data. This is also possible in some rare case as can be observed in Figure. 14 where few distant cars are detected by algorithm but they are not found in ground truth annotation. So this is a feature not a bug of the algorithm.

We evaluate algorithms on several metrics which have been detailed as follows.

A. Intersection over Union (IoU)

This metric measures the accuracy of predicted bounding boxes. First, we measure the area of intersection between

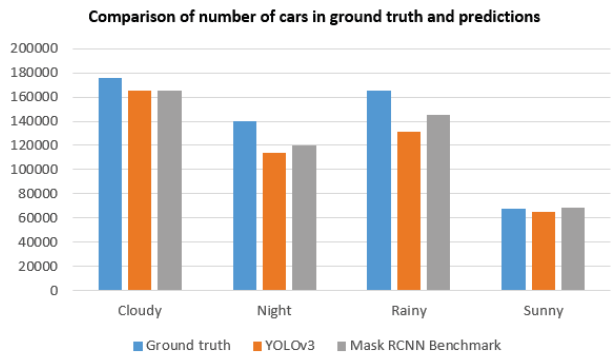

Fig. 12. Comparison of the number of cars in ground truth and predictions

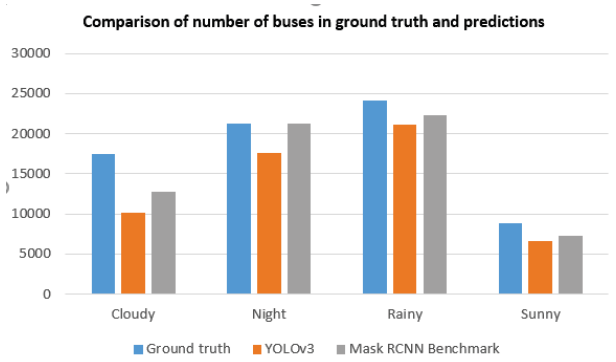

Fig. 13. Comparison of the number of buses in ground truth and predictions

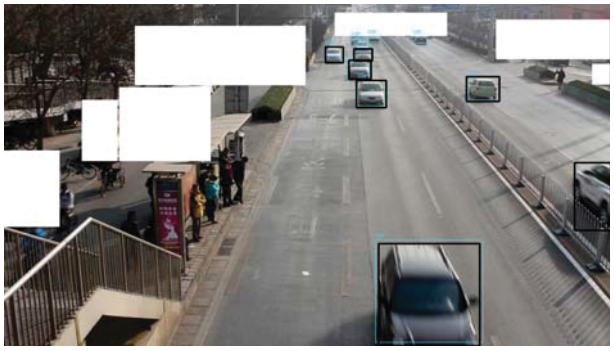

Fig. 14. Block boxes represent ground truth and blue boxes represent detection by Mask RCNN. Few distant cars have been detected by Mask RCNN but not found in ground truth

predicted and ground truth bounding boxes. IoU is computed by taking the ratio of Area of intersection by total area of boxes. We measure Intersection over union values with respect to all ground truth bounding boxes in each frame and take arithmetic mean of all of them. We perform analysis according to the weather. Table IV shows average IoU values for cars and buses using both techniques for different weathers.

B. Precision Recall curve

Precision recall curve is a metric introduced by Pascal VOC paper [41]. In this curve we find precision and recall with respect to every prediction and then plot recall on x-axis and

TABLE IV. AVERAGE IOU VALUES USING BOTH TECHNIQUES ON DIFFERENT WEATHERS

Technique	Vehicle	Cloudy	Night	Rainy	Sunny
YOLOv3	Cars	0.7927	0.7999	0.7784	0.7695
	Buses	0.7404	0.8080	0.7720	0.7610
Mask RCNN	Cars	0.8234	0.8334	0.8150	0.7923
	Buses	0.7788	0.7994	0.8186	0.7648

precision on y-axis. Precision is the ability of the model to identify correct positive predictions.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Total\_predictions} \quad (7)$$

Recall is the ability of object detector to find all relevant ground truths.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Total\_ground\_truth} \quad (8)$$

Hence, in order to calculate this metric, we need to compute true positive (TP) and false positive (FP) values. Figure. 5 shows a block scheme of proposed methodology for calculating TP and FP. Then we compute precision and recall for every detection and from which we obtain precision recall curve. The Figure. 15 shows a PR curve for test portion cloudy weather data for different IoU threshold (0.5, 0.6, 0.7, 0.8 and 0.9).

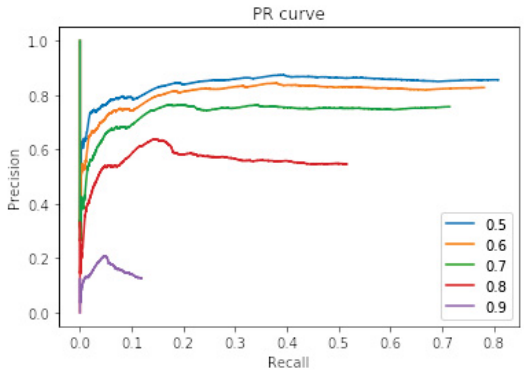

Fig. 15. PR-curve obtained by applying YOLOv3 on Cloudy data on cars for different IoU thresholds (0.5, 0.6, 0.7, 0.8 and 0.9)

C. Average Precision (AP)

Another way to measure the performance of object detectors is Average precision which corresponds to area under the precision-recall curve for a particular class which in our case is cars. This gives us a single number to identify performance instead of a curve. Since the precision recall curve has a very rough zig-zag shape so first this curve is interpolated to form rectangles. The total area of these rectangles corresponds to average precision (AP). Tables V and VI present the average precision values for different weathers at different IoU thresholds using both techniques.

D. Mean Average Precision (mAP)

Mean average precision is simply an arithmetic mean of average precisions for all classes. Following the evaluation protocol suggested by [9], mAP at 0.7 IoU threshold is

TABLE V. AVERAGE PRECISION (AP) USING YOLOV3

Vehicle	Weather	0.5	0.6	0.7	0.8	0.9
Cars	Cloudy data	0.6970	0.6536	0.5430	0.2990	0.0210
	Night data	0.6983	0.6541	0.5430	0.2472	0.0122
	Rainy data	0.5846	0.5440	0.4396	0.1946	0.0087
	Sunny data	0.6918	0.6257	0.5076	0.2401	0.0081
Buses	Cloudy data	0.4432	0.3820	0.2557	0.1130	0.0060
	Night data	0.8319	0.8022	0.7360	0.4967	0.0794
	Rainy data	0.6675	0.5780	0.4440	0.1196	0.0120
	Sunny data	0.5874	0.5014	0.3994	0.1854	0.0052

TABLE VI. AVERAGE PRECISION (AP) USING MASK RCNN BENCHMARK

Vehicle	Weather	0.5	0.6	0.7	0.8	0.9
Cars	Cloudy data	0.6955	0.6588	0.5930	0.4454	0.0830
	Night data	0.7279	0.6840	0.6114	0.3948	0.0560
	Rainy data	0.6278	0.5857	0.5036	0.3368	0.0475
	Sunny data	0.7079	0.6677	0.6040	0.4290	0.0490
Buses	Cloudy data	0.5970	0.5660	0.4882	0.3209	0.0221
	Night data	0.7833	0.7122	0.5612	0.3600	0.0616
	Rainy data	0.6174	0.5780	0.4450	0.3340	0.0020
	Sunny data	0.5926	0.5050	0.4307	0.3023	0.0306

considered as evaluation metric. We take the arithmetic mean of average precision at 0.7 IoU thresholds for all weathers and present our results in Fig. 16. It is absolutely clear that Mask RCNN Benchmark outperforms YOLOv3 in all weather conditions in terms of accuracy.

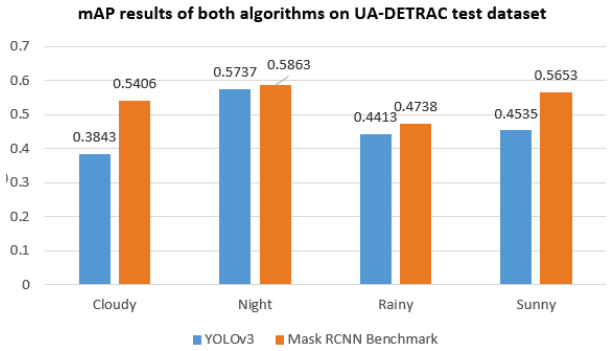

Fig. 16. Mean Average Precision (mAP) results of both techniques on UA-DETRAC test dataset)

E. Tracking Evaluation

The performance of tracker is analyzed by reporting mean absolute error as already done by [11]. Hence, we measure the root mean square error (RMSE) between detected trajectory and ground truth trajectory. For each trajectory point, we calculate the Euclidean distances between point of trajectory and ground truth points of all frames corresponding to the trajectory. The minimum distance corresponds to the loss for this trajectory point. Hence, total loss of each frame is calculated by the sum of losses for all trajectory points and can be described by the following equation:

$$l = \sum_{c=1}^{M} min(\sqrt{(x_c - x_g) + (y_c - y_g)})_{g=1}^{N} \quad (9)$$

where, M is the number of trajectory points in a frame and N is the total number of frames The total loss of tracker is the sum of losses for all frames.

$$L = \sum_{i=1}^{N} l_i \quad (10)$$

The results are reported in Table VII. The values represent the root mean square distance error per detection in units of pixels between detected and ground truth trajectories for different weathers on UA-DETRAC dataset.

TABLE VII. RMS ERROR PER DETECTION IN UNITS OF PIXELS BETWEEN DETECTED AND GROUND TRUTH TRAJECTORIES FOR DIFFERENT WEATHERS ON UA-DETRAC DATASET

Tracker	Detector	Cloudy	Night	Rainy	Sunny
SORT	YOLOv3	15	16	20	16
	Mask RCNN	13	14	15	13

VI. CONCLUSIONS AND FUTURE RECOMMENDATIONS

We implemented the task of trajectory estimation of vehicles in crowded and crossroad scenarios. We adopted tracking by detection scheme in which initially vehicles were detected in each frame and then tracked. We did an analysis of YOLOv3 and Mask RCNN Benchmark using several metrics like IoU, PR curves, Average Precision (AP) and mean Average Precision (mAP). Experiments show that Mask RCNN Benchmark outperforms YOLOv3 in terms of accuracy. The bounding boxes are tracked using SORT tracker. Experiments show that the developed pipeline works well for crowded and crossroad scenarios. All the source code and video demos are available at **https://github.com/hafizas101/Master-s-thesis**

Despite the achieved results, there is still room for further improvements. The performance of detection needs to be improved for vehicles with side view. YOLOv3 and Mask RCNN have been found to be much lesser efficient at recognizing vehicles from side pose in comparison to front or back pose. The speed of our method on our low-end machine is around 2 FPS which is far away from being real-time. Most of the time is taken in the detection process. Hence, accurate and fast detectors are need particularly trained for vehicle detection task to leverage the real potential of tracking by detection scheme.

REFERENCES

[1] R. Abduljabbar, H. Dia, S. Liyanage, and S. A. Bagloee, "Applications of artificial intelligence in transport: An overview," *Sustainability*, vol. 11, no. 1, p. 189, 2019.

[2] W. Li, J. Mu, and G. Liu, "Multiple object tracking with motion and appearance cues," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[3] J. R. B. Del Rosario, A. A. Bandala, and E. P. Dadios, "Multi-view multi-object tracking in an intelligent transportation system: A literature review," in *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. IEEE, 2017, pp. 1–4.

[4] J.-n. Xin, X. Du, and J. Zhang, "Deep learning for robust outdoor vehicle visual tracking," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, pp. 613–618.

[5] R. J. López-Sastre, C. Herranz-Perdiguero, R. Guerrero-Gómez-Olmedo, D. Oñoro-Rubio, and S. Maldonado-Bascón, "Boosting multi-vehicle tracking with a joint object detection and viewpoint estimation sensor," *Sensors*, vol. 19, no. 19, p. 4062, 2019.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[7] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[8] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018, accessed: [27th February, 2020].

[9] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," *arXiv preprint arXiv:1511.04136*, 2015.

[10] K. Kovacic, E. Ivanjko, and H. Gold, "Real time vehicle trajectory estimation on multiple lanes," *sign (Ii- BGk- 1)*, vol. 1, p. 1, 2014.

[11] A. Fedorov, K. Nikolskaia, S. Ivanov, V. Shepelev, and A. Minbaleev, "Traffic flow estimation with data from a video surveillance camera," *Journal of Big Data*, vol. 6, no. 1, p. 73, 2019.

[12] S. R. E. Datondji, Y. Dupuis, P. Subirats, and P. Vasseur, "A survey of vision-based traffic monitoring of road intersections," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 04 2016.

[13] A. Ammar, A. Koubaa, M. Ahmed, and A. Saad, "Aerial images processing for car detection using convolutional neural networks: Comparison between faster r-cnn and yolov3," *arXiv preprint arXiv:1910.07234*, 2019.

[14] J. Sang, Z. Wu, P. Guo, H. Hu, H. Xiang, Q. Zhang, and B. Cai, "An improved yolov2 for vehicle detection," *Sensors*, vol. 18, no. 12, p. 4272, 2018.

[15] K.-J. Kim, P.-K. Kim, Y.-S. Chung, and D.-H. Choi, "Performance enhancement of yolov3 by adding prediction layers with spatial pyramid pooling for vehicle detection," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[16] ——, "Multi-scale detector for accurate vehicle detection in traffic surveillance data," *IEEE Access*, vol. 7, pp. 78 311–78 319, 2019.

[17] D. Impedovo, F. Balducci, V. Dentamaro, and G. Pirlo, "Vehicular traffic congestion classification by visual features and deep learning approaches: a comparison," *Sensors*, vol. 19, no. 23, p. 5213, 2019.

[18] K. Kovacic, E. Ivanjko, and N. Jelušic, "Measurement of road traffic parameters based on multi-vehicle tracking," *arXiv preprint arXiv:1510.04860*, 2015.

[19] S. Seong, J. Song, D. Yoon, J. Kim, and J. Choi, "Determination of vehicle trajectory through optimization of vehicle bounding boxes using a convolutional neural network," *Sensors*, vol. 19, no. 19, p. 4263, 2019.

[20] X. Hou, Y. Wang, and L.-P. Chau, "Vehicle tracking using deep sort with low confidence track filtering," 09 2019, pp. 1–6.

[21] S. Shrestha, "Vehicle tracking using video surveillance," in *Intelligent System and Computing*, Y. C. Yi, Ed. Rijeka: IntechOpen, 2020, ch. 9. [Online]. Available: https://doi.org/10.5772/intechopen.89405

[22] R. López-Sastre, C. Herranz-Perdiguero, R. Guerrero-Gómez-Olmedo, D. Oñoro-Rubio, and S. Maldonado-Bascón, "Boosting multi-vehicle tracking with a joint object detection and viewpoint estimation sensor," *Sensors*, vol. 19, no. 19, p. 4062, Sep 2019. [Online]. Available: http://dx.doi.org/10.3390/s19194062

[23] L. Zhang, Z. Wang, F. Sun, and D. G. Dorrell, "Online parameter identification of ultracapacitor models using the extended kalman filter," *Energies*, vol. 7, no. 5, pp. 3204–3217, 2014.

[24] J. Wang, S. Simeonova, and M. Shahbazi, "Orientation- and scale-invariant multi-vehicle detection and tracking from unmanned aerial videos," *Remote Sensing*, vol. 11, no. 18, p. 2155, Sep 2019. [Online]. Available: http://dx.doi.org/10.3390/rs11182155

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[26] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[27] D. Uchechukwu, A. Siddique, A. Maksatbek, and I. Afanasyev, "Ros-based integration of smart space and a mobile robot as the internet of robotic things," in *2019 25th Conference of Open Innovations Association (FRUCT)*. IEEE, 2019, pp. 339–345.

[28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[29] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2980–2988.

[32] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017, pp. 1–6.

[33] E. Bochinski, T. Senst, and T. Sikora, "Extending iou based multi-object tracking by visual information," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[34] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[35] R. G. Brown, P. Y. Hwang *et al.*, *Introduction to random signals and applied Kalman filtering*. Wiley New York, 1992, vol. 3.

[36] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[37] S. Hossain and D.-j. Lee, "Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with gpu-based embedded devices," *Sensors*, vol. 19, no. 15, p. 3371, 2019.

[38] N. Peri, P. Khorramshahi, S. S. Rambhatla, V. Shenoy, S. Rawat, J.-C. Chen, and R. Chellappa, "Towards real-time systems for vehicle re-identification, multi-camera tracking, and anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[40] K. Guan, "Vehicle detection and tracking," https://github.com/kcg2015/Vehicle-Detection-and-Tracking, 2019.

[41] S. Vicente, J. Carreira, L. Agapito, and J. Batista, "Reconstructing pascal voc," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 41–48.