Journal of Big Data Analytics in Transportation
https://doi.org/10.1007/s42421-021-00042-3

ORIGINAL PAPER

# Deep Learning Techniques for Vehicle Trajectory Extraction in Mixed Traffic

**Rohan Dhatbale**[1] · **Bhargava Rama Chilukuri**[1]

Received: 19 August 2020 / Revised: 26 December 2020 / Accepted: 13 April 2021


**Abstract**
Vehicle trajectories provide very useful empirical data for studying traffic phenomena such as vehicle following behavior, lane changing behavior, traffic oscillations, capacity drop, safety analysis, etc. However, there are a very limited number of studies on extracting trajectory data from mixed traffic and for congested conditions. This paper presents a deep learning-based framework to extract vehicle trajectories in mixed traffic under both free-flow and congested conditions. The popular YOLOv3 deep learning architecture is used and trained on a hybrid dataset generated from two different sets of frames with different scales and orientations. The anchor boxes for vehicle detection and classification are customized to improve accuracy and efficiency. The SORT algorithm is used to track the identified vehicles and the extracted trajectory data are benchmarked with a popular trajectory extraction portal that showed that the proposed model performs well for trajectory extraction. The paper also presents a methodology based on numerical integration techniques to impute missing trajectory data. Finally, the trajectory data obtained from the adjacent road sections are aligned and scaled to the real-world coordinates using coordination transformation and error correction methods to make it useful for research purposes. The extracted trajectories show remarkable accuracy with approximately 0.25–0.35 m of precision. It is expected that these trajectories capture traffic and driving behavior phenomena for a better understanding of mixed traffic conditions.

**Keywords** Deep learning · Image processing · Vehicle trajectory extraction · Mixed traffic conditions · YOLOv3

## Introduction

With the ongoing traffic growth in urban road networks, the management of ever-increasing traffic volumes and congestion levels is one of the most critical challenges faced by engineers and managers. To develop suitable strategies, the study of traffic phenomena such as drivers' characteristics, lane changing behavior, traffic oscillation, capacity drops, and crash potential are very important. The vehicle-level data, such as trajectories of the vehicles on the road, provides very useful empirical data to carry out these studies. Driving behavior models capture drivers' maneuvering decisions in different traffic conditions, which are also an essential component of microscopic traffic simulation applications. Therefore, vehicle trajectories immensely help in understanding traffic and driving behavior for developing traffic management strategies.

Next-Generation Simulation (NGSIM) (2005) is a popular and publicly available vehicle trajectory database collected for homogeneous and lane-based traffic conditions. NGSIM data were collected by an automated vehicle tracking system using surveillance cameras. Subsequently, manual corrections for false positive and false negative errors were made (e.g. Montanino and Punzo 2013; Coifman and Li 2017). Venkatesan Kanagaraj et al. (2016) is a publicly available vehicle trajectory database representing heterogeneous traffic conditions. The data was collected with videos from a vantage position, but this data only involves moderately congested conditions.

To overcome some of these limitations, videos from Unnamed Aerial Vehicles (UAVs) or videos recorded from a vantage position are an attractive alternative. UAVs exhibit many advantages, such as ease of deployment, high mobility, large visual range, reasonable uniform scale, etc. UAVs can record a road with different lengths by adjusting flying

✉ Bhargava Rama Chilukuri
bhargava@iitm.ac.in

Rohan Dhatbale
rohandhatbale@gmail.com

[1] Department of Civil Engineering, Indian Institute of Technology Madras, Chennai, India 600036

Published online: 04 May 2021

altitude to fulfill different data collection requirements. Compared with pole-mounted cameras, the video recorded by UAVs has less occlusion of vehicles in a lane. It could measure a vehicle's position more accurately from the top view by hovering at the desired altitude.

While there have been some studies on using aerial imagery for traffic data extraction, they have been mostly performed for homogeneous conditions. Trajectory extraction in heterogeneous traffic conditions is challenging due to the heterogeneity in vehicle classes and lack of lane discipline. Recently, deep learning approaches, like convolutional neural network (CNN), and methods based on Transfer Learning have been suggested for traffic data extraction under homogeneous conditions for improved accuracy. Therefore, this study focuses on exploring the use of deep learning approaches on videos recorded from a vantage point for vehicle detection and trajectory extraction under mixed traffic conditions.

The organization of the paper will be as follows. The next section presents a review of literature on vehicle detection and classification, vehicle tracking, and trajectory extraction using conventional methods as well as recent machine learning and deep learning techniques. Section "Vehicle identification and classification" describes the implementation of deep learning techniques for vehicle identification and classification. Section "Vehicle trajectory extraction" presents the methodology for automatic vehicle trajectory extraction. Finally, Sect. "Discussion" discusses the highlights and conclusions of this study.

## Literature Review

This section presents a review of the literature on vehicle detection and classification, vehicle tracking methods, and trajectory extraction using pole-mounted and UAV-based video recordings.

Traditionally, vehicle detection and classification are performed in a series of steps: pre-processing, feature extraction, detection, and finally, classification. Some common techniques used in pre-processing are image registration, geo-registration, sliding widow search algorithm, etc. Wang et al. (2012) used image registration and geo-registration techniques to pre-process the video. These techniques were primarily used to remove non-uniform pixel-level noise-induced due to instability and vibration. Leibe et al. (2005), Wang et al. (2012, 2014), Milan et al. (2013) used a sliding window search algorithm for the generation of candidate regions. Henriques et al. (2014) proposed Region Proposal Network (RPN) and Bay et al. (2008) proposed a CNN-based method called DeepBox to generate candidate regions with deep learning features for region proposal. Region proposal is used to decide 'where' to look in the image or frame to reduce the computational cost of the overall inference process. Khan et al. (2017) used video trimming, image rectification, and region-of-interest masking in the pre-processing stage. Image distortions such as fish-eye and darkened-edges effects caused by the type and settings of lens used are removed or minimized to prepare the video for the processing and analysis phase. Song et al. (2019) proposed a vision-based vehicle detection and counting system by dividing the roadway into a remote area with no vehicles and a proximal area. The proximal area is fed into the YOLOv3 framework to locate the vehicle and classification.

For the feature extraction and detection of vehicles, researchers have used techniques such as corner detection, edge detection, optical flow analysis, local feature points, etc. Tsai et al. (2007) used edge detection along with normalized color maps to detect the vehicles. They proposed a new color transform model for quickly locating possible vehicle candidates. After getting possible vehicle candidates, corner detection and edge detection are used to detect the vehicle. Cheng et al. (2012) used the Dynamic Bayesian Network (DBNs) to detect vehicles. The color transform technique was used to separate vehicle colors from non-vehicle colors. Then, Canny edge detection was used to find the edges of the vehicles for constructing DBN for the classification of vehicles and non-vehicle regions. Choi et al. (2011) and Garcia et al. (2012) used optical flow to detect the vehicles on the road. The former used optical features along with Haar-like features to detect the vehicles whereas the latter used optical flow along with the radar data to detect the vehicles. Leibe et al. (2005), Miao et al. (2011), adopted Histogram Oriented Gradient features for vehicle detection. Wang et al. (2012) made use of Scale Invariant Features Transform descriptors, whereas Wang et al. (2014) used Haar-like features and Local Binary Pattern for vehicle detection.

Leibe et al. (2005), Miao et al. (2011), Milan et al. (2013) employed AdaBoost classifier for the classification. Wang et al. (2012) used SVM along with handcrafted features for detection and classification. Milan et al. (2013) proposed a method by subtracting the background colors of each frame and then refined vehicle candidate regions by enforcing the size constraints of vehicles. Wang et al. (2012) proposed a moving vehicle detection method based on cascade classifiers. Also, multi-scale sliding windows were generated at the detection stage. However, their method resulted in a lot of missed detections on rotated vehicles. Ozkurt et al. (2009) used a neural network for vehicle classification and vehicle counting from the video. In recent years, deep learning approaches based on CNN have shown impressive performance in vehicle detection. Especially, Region-based Convolutional Neural Networks (R-CNN) in 2014, Fast R-CNN in 2015, and Faster R-CNN in 2017 have been proposed to precisely locate objects in an image. Ren et al. (2015)

used Faster-RCNN for the calculation of traffic density. Adu-Gyamfi et al. (2017) used Deep Convolutional Neural Networks for vehicle classification. Onoro et al. used two different CNN variants namely counting CNN and Hydra CNN for vehicle counting and traffic density prediction. Xu et al. (2017) applied Faster R-CNN for vehicle detection from UAV images, and Faster R-CNN showed better results than conventional CNN with a sufficient amount of computation cost and training samples. Chen et al. (2018) proposed a modified Faster R-CNN for small-sized vehicles from UAV images. Chen et al. (2020) developed a methodology to extract high-resolution vehicle trajectories from UAV. The study involved the detection of a vehicle in the target region and tracking using kernelized correlation filter. After this, a transformation of coordinates from cartesian coordinates to Frenet coordinated is carried out to extract the usable raw trajectories. More recently, Haghighat et al. (2020) gave a comprehensive review of Deep Learning applications in Intelligent Transportation Systems. This paper presented a comprehensive overview of the DL techniques for ITS applications, their advantages, and disadvantages.

In the area of vehicle tracking, Koller et al. (1994) proposed a methodology to deal with occlusion by employing explicit occlusion reasoning coupled with Kalman filters. Rothrock and Drummond (2000), Danescu et al. (2009), and Hue et al. (2002) utilized a sequential Monte Carlo Method and Particle filter algorithm, to speed up the process of tracking and to accommodate the non-gaussian nature of the problem. Yilmaz et al. (2006) carried out a detailed survey of numerous tracking approaches presented in the past. Rodríguez-Canosa et al. (2012) developed a real-time method to detect and track moving objects from UAVs using artificial optical flow techniques. Xu et al. (2017) introduced a vehicle detecting and tracking method for low-angle camera video using a cellular neural network to subtract the background and to refine the detection results with optical flow.

Several researchers have attempted to extract vehicle trajectories for traffic analysis purposes. Kim et al. (2005) developed an extensive trajectory dataset using Next-Generation Simulation. Oh et al. (2009), St-Aubin et al. (2013), Li et al. (2014) applied different image processing techniques to the video recorded using a pole-mounted camera to extract multi-vehicle trajectories. Oh, et al. (2009) presented an automated video analysis system that can detect and track road users in traffic and classify them as pedestrians or motorized road users. St-Aubin et al. (2013), proposed a methodology for the analysis of driving behavior, trajectory interpretation, and conflict measures using computer vision. Li et al. (2014), proposed a system based on identifying cyclist helmets from video footage. Gao et al. (2014) presented a methodology for the automatic extraction of vehicle trajectories. However, in their approach, the user initially has to manually select the vehicle to be tracked. Gao et al. (2014), Guido et al. (2014), Apeltaeur et al. (2015), Barmpounakis et al. (2016), Kim et al. (2019) used UAV videos to extract vehicle trajectories. Kim et al. (2005) used the background subtraction algorithm that uses a corner feature tracking and a grouping algorithm for vehicle tracking. Jodoin et al. (2014) proposed a tracking method that has been specifically designed to track the various road users in urban environments using background subtraction. Aubrey et al. (2019) proposed a fully automatic and scalable framework for the extraction of vehicle trajectories using a single modular traffic camera mounted at the roadside. The authors of the recently released Mask R-CNN (He et al. 2017) used it to detect the vehicles on the frame by frame. Mask RCNN provides object detection with an instance mask of the object, a 2D bounding box, and an instance type which was used to estimate the ground position of the vehicle and the measurement to track the association. The frame by frame detections was then grouped into vehicle trajectory using IoU (Intersection over Union) method and smoothened.

While the studies discussed earlier dealt with videos from pole-mounted cameras, some studies used aerial imagery to extract vehicle trajectories. Li et al. (2009) proposed a method to detect and track vehicles on the highway based on UAV video. They claimed high levels of accuracy, but the background traffic volume was very low and dispersed. Cheng et al. (2012) designed a vehicle detection framework that preserves the advantages of the existing works and overcomes their drawbacks. The detection part in the proposed framework was based on pixel-wise classification, whereas features were extracted in a neighborhood region of each pixel for high levels of computational efficiency. However, the traffic conditions were only mildly congested. Jırı et al. (2015) proposed an approach for simultaneous detection and tracking of vehicles at intersections with low levels of congestion. However, a manual effort was required for improved accuracy of results. Wang et al. (2016) introduced a new vehicle detecting and tracking system based on consecutive frames to generate a vehicle's dynamic information, such as positions and velocities. The system uses multiple consecutive images to increase the system accuracy of detecting and tracking vehicles under homogeneous and low traffic volume conditions. Khan et al. (2017) proposed a framework for automated UAV video processing from four-leg intersections. The system was tested on UAV videos from altitudes of 80 m and 60 m. The computer vision-based algorithm showed that the accuracy of the calibration process is critical for minimizing errors. Kim et al. (2019) proposed and compared two learning-based frameworks for detecting vehicles; the aggregated channel feature-based on human-made features and Faster R-CNN-based on data-driven features. It was argued that road signs, shadows, and trucks, were the main reason for errors in

the extracted trajectories. In the tracking process, these not only became the false positives of trajectories but also caused mismatches with other trajectories.

It is evident from the literature that most of the image processing techniques are focused on imagery from a pole-mounted camera and only a limited number of studies focused on aerial imagery from UAVs. However, even, the limited studies based on aerial imagery have focused on homogeneous and lane-based traffic. Strategies to overcome the challenges associated with image processing in heterogeneous and lane less traffic are not sufficiently explored. To overcome this limitation, this paper presents a methodology based on the customization of the emerging deep learning techniques for vehicle identification and trajectory extraction in mixed traffic conditions. Some of the other highlights of this paper include: (1) a hybrid training technique to take advantage of the scale invariance of the YOLOv3 framework, (2) customization of anchor boxes for improved accurate and computational efficiency, (3) a numerical integration methodology for trajectory imputation, and (4) a novel scaling technique to minimize error in transforming image coordinates to ground coordinates.

## Vehicle Identification and Classification

This section presents details on the data collection effort, the deep learning architecture used in this study, hybrid training methodology, customization of anchor boxes, and results on detection and classification.

### Data Collection

A three-lane Sardar Patel Road near the Madhya Kailash Intersection in Chennai, India was chosen for the study. Two adjacent road sections, called view2 (70 m) and view3 (50 m), were recorded using separate cameras for 1 h 50 min from a vantage point. As can be seen from Fig. 1, the road in the two views is not aligned and is also different. The southwest end of the road in view2 joins the east end of the road in view 3 (see Fig. 1) giving approximately 100 m continuous section. Note that the last street light pole in view2 is also visible as the first street light pole in view3. Both the videos resemble UAV recorded aerial imagery as these were recorded from the top of an adjacent building giving a plan view of the road with very minimal occlusion of smaller vehicles due to larger vehicles.

### Selection of a Deep Learning Model

"You Only Look Once" is an algorithm that uses CNN for object detection (Redmon et al., 2016). It is one of the faster object detection algorithms currently available and has shown to perform well for real-time detection, without loss of too much accuracy.

Redmon and Farhadi (2018), authors of YOLOv3, compared various state-of-art deep learning architectures and claimed that YOLOv3 performs better than other architectures in terms of accuracy and mean Average Precision (mAP). Also, YOLOv3 addresses issues faced by its previous version YOLOv2 such as problems in detecting smaller objects, a lesser number of anchor boxes, etc. Later, Jonathan (2019) carried out an independent comparative study between state-of-art deep learning architectures like Faster R-CNN, SSD, and YOLOv3 in terms of accuracy and speed. The results showed that Faster R-CNN has 70%, SSD@300 has 72.4% and SSD@512 has 74.9% accuracy respectively. However, YOLOv3@416 and YOLOv3@512 showed 76.8% and 78.6% accuracy. Moreover, the lowest and highest speed recorded by Faster R-CNN was 5 fps and 17 fps, SSD was 22 fps, and 59 fps, but YOLOv3 recorded 40 fps and 91 fps. From these results, it was concluded that YOLOv3 outperforms Faster R-CNN and SSD in both accuracy as well as speed. Moreover, literature shows that one of the strengths of YOLOv3 is scale invariance on object detection (Redmon

**Fig. 1** View2 and View3 respectively

and Farhadi, 2018). This means that objects with similar features, but at different scales will still be identified. This feature greatly helps in the context of heterogeneous traffic because there are several vehicle classes and the scaling errors can have a significant impact on accurate detection; for example, a heavy truck may be identified as a light commercial vehicle at a different scale. Based on these reasons, YOLOv3 is selected as the deep learning architecture for this study. The popular Darknet platform developed by the authors of YOLOv3 (Redmon and Farhadi, 2018), is used in this study. It is an open-source neural network framework written in C and CUDA to provide a platform for using object detection architectures like YOLOv3.

## Training YOLOv3 for Custom Object Detection

YOLOv3 is pre-trained on the *COCO trainval* dataset with 80 different classes for prediction. To reduce training time, transfer learning techniques with default weights were used for training the model for custom vehicle classes. A total of 236 frames, 118 from each view (view2 and view3), were extracted at an interval of 10 s such that no two consecutive frames have the same vehicle repeated for an unbiased training dataset. The number of instances for some classes such as truck, bus, and lcv (Light Commercial Vehicle) after annotation were very less (less than 200), which may not be sufficient for training the model. Hence, data augmentation is carried out to increase the size of the dataset. In data, augmentation frames with trucks, buses, and lcvs were filtered, and common augmentation techniques like randomly applying Gaussian Blurr, changing contrast, and rotation were applied. After augmentation, a 1319 frame dataset was created. Table 1 shows the detailed dataset information. The augmented dataset was annotated using *LabelImg* annotating tool (Tzutalin, 2015). This tool helps in annotating the image/frame and creating the *train.txt*, *test.txt*, *obj.names*, and *obj.data* files necessary for training the model. The *train.txt* and *test.txt* files have the path of all the images for training and testing respectively. The *obj.names* file has the names of all the classes of our interest on which model is going to be trained and *obj.data* file has information like the number of classes, path to train and test dataset, the path to the backup folder where the trained weights can be saved while training the model. The configuration file is customized for the number of filters in the YOLOv3 layer, parameters like *batch_size*, *subdivision*, *max_batches* according to the GPU power to avoid memory issues while training.

One of the novelties of this work is the use of frames from different views, view2 and view3, to create a hybrid training dataset. Initially, separate training datasets with frames from view2 and view3 were created and tested on the respective views. Later, it was decided to create a combined training dataset based on frames from both the views and test on images from both the views. This was done to check how well YOLOv3′s scale-invariant feature works on heterogeneous traffic. Table 1 shows the class-wise distribution of objects in training, validation, and test datasets.

**Table 1** Hybrid training dataset

| Quantity | Training | Test | Total |
|---|---|---|---|
| Frames | 1319 | 38 | 1357 |
| Cars | 16,368 | 392 | 16,760 |
| Light commercial vehicle | 963 | 29 | 992 |
| Two-wheeler | 10,430 | 264 | 10,694 |
| Truck/tanker | 1224 | 25 | 1249 |
| Three-wheeler | 2934 | 93 | 3027 |
| Bus/mini-bus | 1179 | 29 | 1208 |

## Custom Anchor Boxes Generation for YOLOv3

YOLOv3 uses 9 different anchor boxes for each cell. Since they are trained on the COCO dataset which has over 80 different classes, the default anchor boxes are suitable for the COCO dataset. Figure 2 shows the anchor boxes given by YOLOv3. While making predictions, it divides the image in the S × S grid and for each cell in the grid, it uses 9 anchor boxes having fixed width and height to predict objectness along with the confidence of each anchor box for each cell.

When these default anchor boxes were used as-is on the view2 and view3 videos, it gave a mean Intersection over Union (IoU) of 69.45%. Moreover, it took a significant amount of time and computational resources to converge. To make this process more efficient, images from the study site were studied to identify suitable anchor box sizes. It was found that none of the vehicles are as large as anchor box 1 or 2 shown in Fig. 2a. Therefore, these anchor boxes are deemed to be redundant for the study.

K-means clustering is used to identify good candidate anchor boxes. However, according to Redmon and Farhadi (2018), using the standard Euler distance metric minimizes error for large bounding boxes but not for smaller boxes as Euclidean distance metric causes larger boxes to generate more errors than smaller boxes. So, to overcome this problem, IoU is used as a metric to identify suitable anchor boxes. Figure 3 shows the plot of width vs the height of the bounding boxes accompanying all vehicles in the extracted dataset. The width and height (in pixels) of the bounding boxes are normalized using the actual width and height of the image (in pixels).

The constant slope in Fig. 3a for class 'car' shows that most of this class vehicle have fixed aspect ratio, whereas, rest of the classes, namely 'tw', 'lcv', 'truck', and 'bus' have diversified dimensions; e.g., larger trucks vs. smaller trucks. Based on this observation, separate custom anchor boxes for

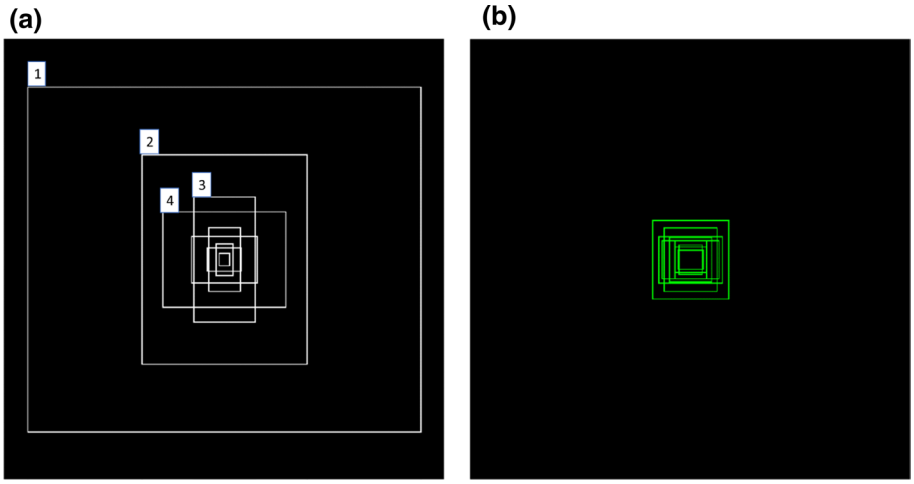



**Fig. 2** **a** Default anchor boxes in YOLOv3 and **b** Customized anchor boxes used in this study

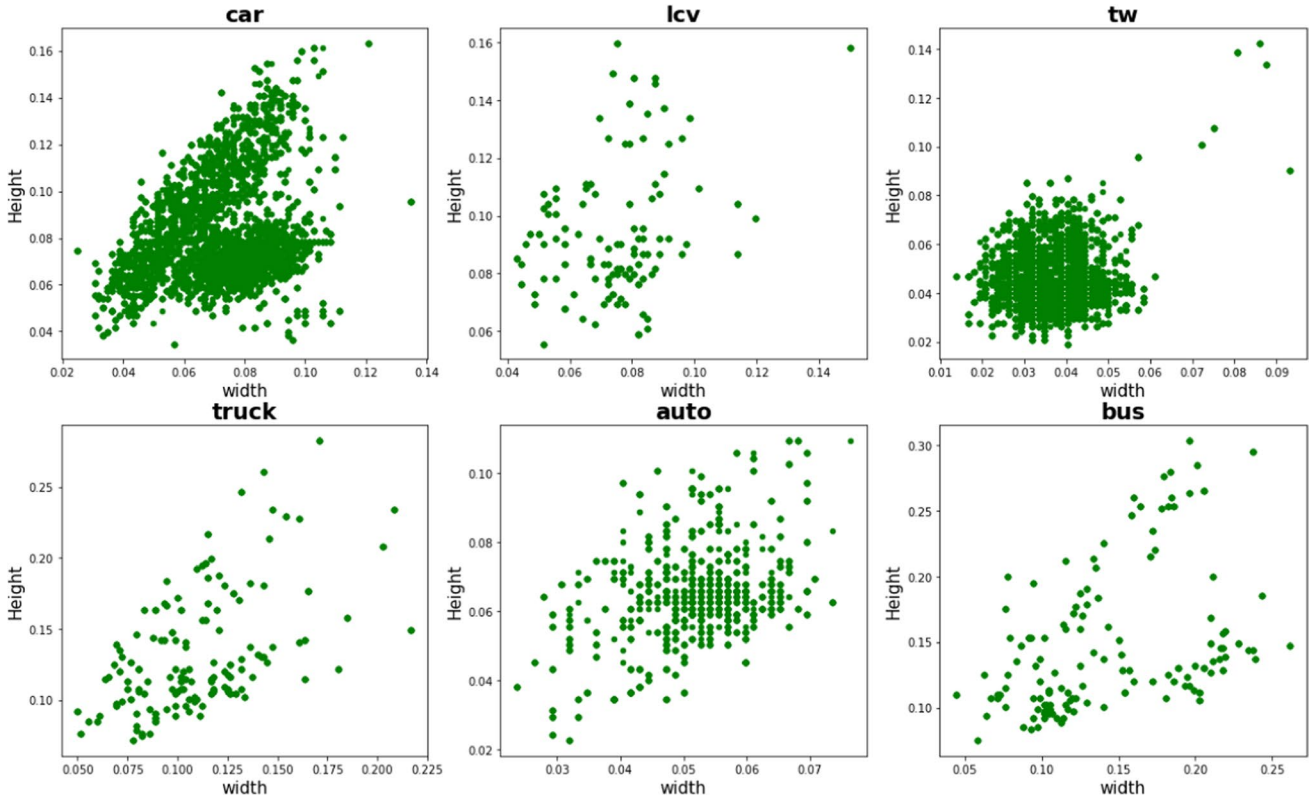



**Fig. 3** Width vs. height scatter plot

this dataset were generated. It helped in reducing the training time and improve the IoU. As can be seen from the plot shown in Fig. 4, IoU increases with an increase in the number of bounding boxes. This signifies that if we increase the initial number of anchors or bounding boxes to be predicted, it increases IoU. However, one should note that the higher the number of boxes, the higher the computational cost.

As shown in Fig. 4, nine anchor boxes give a mean IoU of 83.17% that is deemed sufficient for the model. Thus, the mean IoU has increased from 69.45% to 83.17% with the customized anchor boxes. Also, the training time of the model decreased significantly from more than 28 h (with default anchor boxes) to around 18 h (with custom anchor boxes). Moreover, it took fewer iterations to converge the error within the threshold. Hence custom anchors are used while training the model. Table 2 shows the training parameters used.

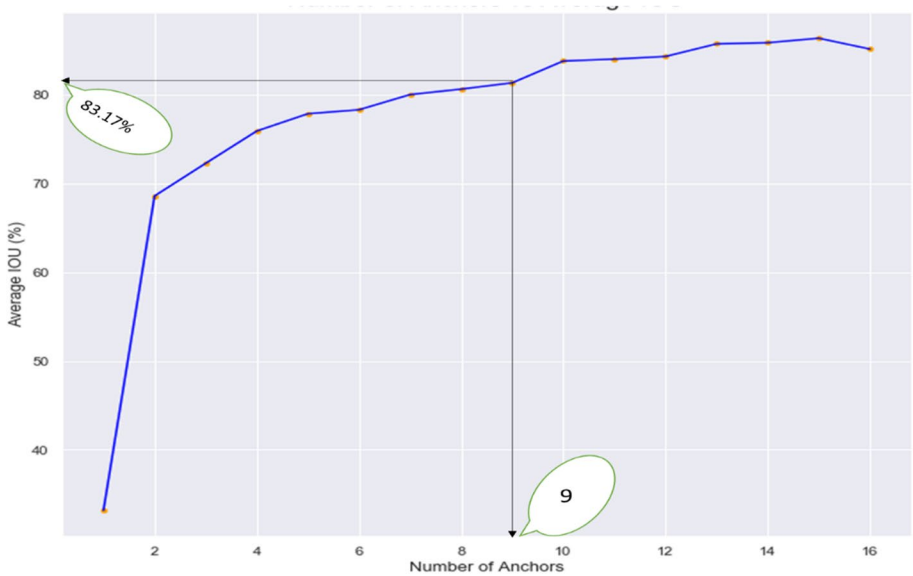



**Fig. 4** Number of anchors vs mean IoU

**Table 2** Yolov3 training parameters

| Parameters | Values |
|---|---|
| img_size (pixels) | 608*608 |
| Batch | 64 |
| Subdivision | 16 |
| Learning_rate | 0.001 |
| Classes | 6 [car, lcv, tw, truck, auto, bus] |
| Filters | 33 |
| Max_batches | 12,000 |
| Custom anchors boxes | ([22, 28], [60, 44], [50, 61], [73, 75], [54, 37], [31, 36], [31, 24], [24, 18], [40, 43]) |

## Detection and Classification Results

To evaluate the performance of detection and classification, four standard metrics are used; precision, recall, F1 score, and quality. These metrics are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = 2 * \text{Precision} * \text{Recall}/(\text{Precision} + \text{Recall})$$

$$\text{Quality} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

where TP (True Positives) shows the number of vehicles successfully detected by the algorithm. FP (False Positives) indicates the number of non-vehicle objects that are falsely detected as vehicles. FN (False Negative) indicates the number of vehicles that the algorithm did not recognize as a vehicle.

The precision of a vehicle class signifies the probability that the vehicle truly belongs to that vehicle class. Recall measures the ability of the algorithm to detect all the instances of vehicles in the image. The F1 score, which is a harmonic mean of precision and recall gives a global measure of the algorithm's robustness (the ability to extract all the instances of a vehicle class and to not falsely classify it). Finally, quality is very similar to precision, but also includes the effect of false negatives.

Figure 5 shows a snippet from model testing using the Google Colab platform (Bisong, 2019). As can be seen, the model can detect, localize, and classify the vehicles in the frames from view2 as well as view3 with the hybrid testing dataset.

Table 3a and b show the number of instances and % detection for each class along with the confusion matrix. It can be seen that the detection rates of lcvs, trucks, and buses are 100% and the same for the other vehicles was over 98%. The confusion matrix in Table 3b shows that truck and lcv have a greater number of false detections compared to any other class. Figure 6 shows the evaluation metrics for the model.

The evaluation metrics in Fig. 6 show that the algorithm has a high precision rate of over 95% for cars, two-wheelers, and Autos, and for buses, it is almost 90%. Recall results, F1 score, and quality score shows that the model consistently

performed well for cars, two-wheelers, autos, and buses. Trucks and lcvs have low values for all metrics compared to other vehicle classes. It is conjectured that differences in orientation in view2 and view3 may be the reason for this. It was found that features of trucks in orientation may have been very similar to the features of LCVs in another view. This conclusion was made because, with a separate training set for each view, the detection of trucks was better than with the hybrid training dataset. Also, a smaller number of instances of truck (145) and lcv (174) in the training data compared to the car (2666), two-wheeler (1850) and three-wheeler (474) may have resulted in not sufficiently capturing

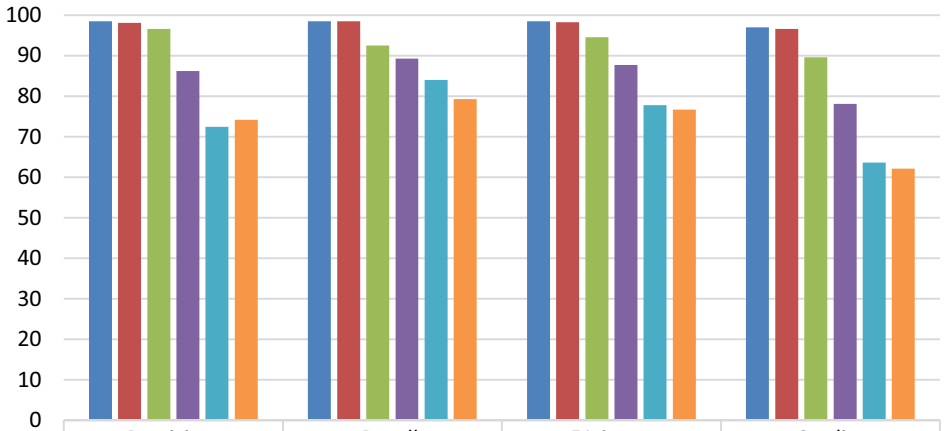
**Fig. 5** Detection result examples

**Table 3** **a** Percentage detection and classification and **b** Confusion Matrix

| | instances | detections | % detection |
|---|---|---|---|
| **car** | 392 | 386 | **98.46** |
| **lcv** | 29 | 29 | **100.00** |
| **tw** | 264 | 261 | **98.86** |
| **truck** | 25 | 25 | **100.00** |
| **auto** | 95 | 93 | **97.89** |
| **bus** | 29 | 29 | **100.00** |

| | | Predicted | | | | | |
|---|---|---|---|---|---|---|---|
| | | **car** | **lcv** | **tw** | **truck** | **auto** | **bus** |
| Actual | **car** | 381 | 2 | 0 | 3 | 1 | 0 |
| | **lcv** | 1 | 23 | 0 | 4 | 0 | 1 |
| | **tw** | 1 | 0 | 257 | 0 | 2 | 1 |
| | **truck** | 0 | 2 | 0 | 21 | 0 | 2 |
| | **auto** | 1 | 1 | 5 | 0 | 86 | 0 |
| | **bus** | 2 | 0 | 0 | 1 | 0 | 25 |

**Fig. 6** Comparison of evaluation metrics




| | Precision | Recall | F1 Score | Quality |
|---|---|---|---|---|
| car | 98.4496124 | 98.4496124 | 98.4496124 | 96.94656489 |
| tw | 98.09160305 | 98.46743295 | 98.2791587 | 96.61654135 |
| auto | 96.62921348 | 92.47311828 | 94.50549451 | 89.58333333 |
| bus | 86.20689655 | 89.28571429 | 87.71929825 | 78.125 |
| truck | 72.4137931 | 84 | 77.77777778 | 63.63636364 |
| lcv | 74.19354839 | 79.31034483 | 76.66666667 | 62.16216216 |

the distinct features of lcvs and trucks. Table 4 shows the class-wise IoU results of the model.

Results in Table 4 show that the model performed very well to identify and locate the vehicles accurately in the frame. For other classes, the two-wheeler has a lower IoU (69.36%). It is conjectured that the smaller size of two-wheeler w.r.t. other vehicles and manual annotations that may have resulted in inconsistent annotation could be reasons for an inferior IoU performance for two-wheelers. However, the IoU values in this study outperform typical values observed in the literature, which reported average values in the range of 60%.

Overall, the results show that the hybrid training dataset and customized anchor boxes used in this study have resulted in a satisfactory performance for the detection and classification of vehicles. The next section describes the steps involved in vehicle trajectory extraction.

## Vehicle Trajectory Extraction

This section begins with a description of the methodology used to track the vehicles and extract trajectories. This is followed by trajectory validation and post-processing steps such as trajectory imputation, scaling, etc.

After detection and classification are done, the next task is to assign an individual ID for each detected vehicle and then track those vehicles to extract trajectories. The SORT algorithm (Bewley et al., 2016) is used to track the identified vehicles across the frames. The algorithm contains 8 variables (*cx, cy, ar, h, cx`, cy`, ar`, h`*) where (*cx, cy*) are centers of the bounding boxes, *ar* is the aspect ratio, and *h*, the height of the image. The other variables are respective derivatives of the above-stated variables. The algorithm uses Kalman filters to track the above variables in successive frames which are primary variables for the bounding box. The Kalman filter helps to factor in the noise in the detection and uses a prior state in predicting a good fit for bounding boxes. For each detection, a "Track" is created, that has all the necessary time-series information. Also, the algorithm has the provision to delete the tracks of the vehicles which have left the frame. To eliminate duplicate tracks, the algorithm has a minimum number of detections threshold for the first few frames. Figure 7 shows sample trajectories extracted using the model.

**Table 4** Result of class-wise IoU values

| Class | IoU (%) |
|---|---|
| Car | 80.79 |
| lcv | 73.55 |
| Two-wheeler | 69.36 |
| Truck | 80.87 |
| Auto | 83.09 |
| Bus | 77.62 |

To benchmark the tracking results, 50-s video clips from view2 and view3 videos are extracted, which had 1269 frames each. Table 5 shows the data extracted from the study area for a 50 s period.

Them annually extracted trajectories of the vehicles are matched with those tracked by YOLOv3. Figure 8 shows the plot of the ground truth trajectory and the automatically extracted trajectory of the same vehicle. RMSE metric is used to evaluate the trajectory extraction process using 40 different vehicles (20 from each view). The mean RMSE was found to be 1.4247 px, which approximately translates to 12 cms.

Further for detailed analysis and evaluation obtaining ground truth trajectories is a cumbersome task hence the trajectories are benchmarked with trajectories obtained from the DataFromSky (DFS 2020) platform. DataFromSky is a popular portal that claims to use Artificial Intelligence and machine learning techniques for fully automated video data extraction. Among 73 (107) vehicles observed in view2 (view3), DFS tracked 58 and 104 vehicles respectively. Thus, it is clear that the model proposed in this paper performed satisfactorily for vehicle detection and tracking.

However, it can be observed from Fig. 7 that some of the tracked vehicles have broken trajectories. This is mainly due to the occlusion of vehicles by the roadside trees and rarely occlusion due to large vehicles (see Fig. 1a). So, the trajectories are post-processed to impute missing data.

### Trajectory Imputation

Literature shows that data imputation is generally performed using numerical interpolation techniques (Gould et al. 2016). However, this technique is purely data-driven and is not grounded on physical principles. To address this gap, we use numerical integration techniques in this paper. We extract acceleration and velocity at both ends of a trajectory discontinuity and use numerical integration techniques such as the Euler-Cromer method, Midpoint method, Velocity Verlet method, and Beeman method to impute the internal points. Additionally, the regression method is employed for some trajectories. The data is imputed from one end and matched for the position, speed, and acceleration at the other end. It was observed that different methods performed well for different trajectories. The best performing method is chosen to complete the trajectory. Figure 9 shows a sample imputation case for vehicle id 1397 from view2. It can be seen that the Beeman method performed better than other methods. Any mismatch at the joining was locally smoothed using the moving average method (see Fig 9). Another variant one could also attempt is to impute from both ends and match the position, speed, and acceleration in the middle.

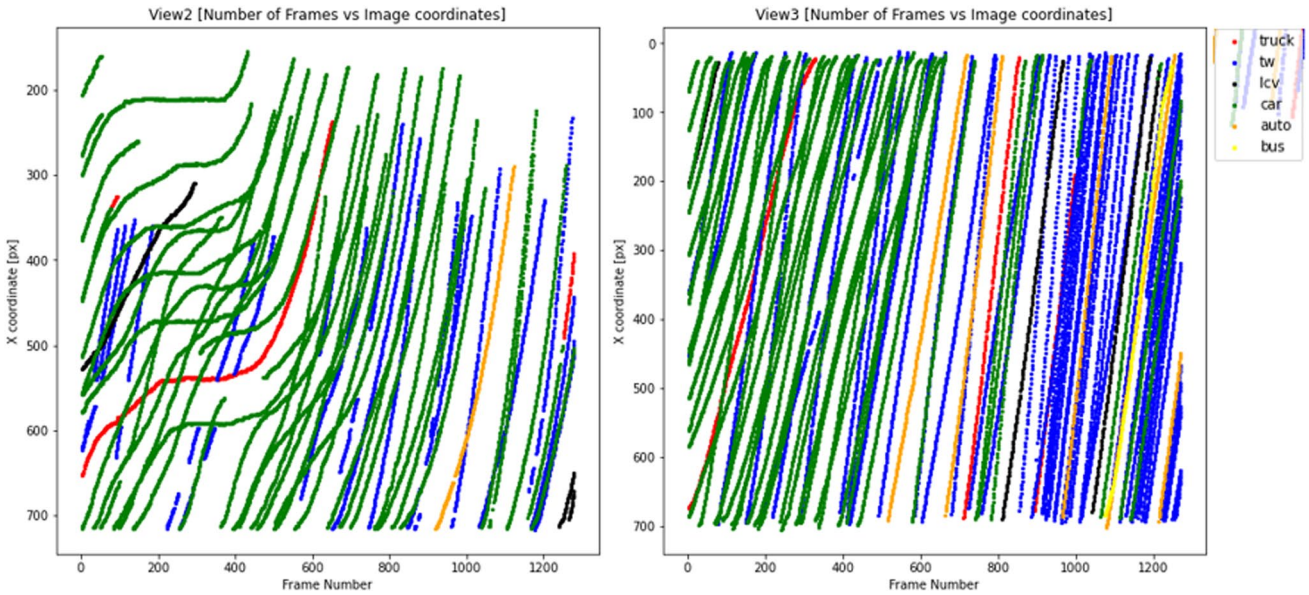



**Fig. 7** Sample vehicle trajectories extracted from view2 and view3

Figure 10 shows that the numerical integration methods imputed the missing data effectively and the trajectories look smooth and realistic.

## Tracking Analysis

The tracking data for the vehicles in view2 and view3 is evaluated using Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA). MOTP is the total localization or position error of detected vehicles' overall frames divided by the total number of matches made. MOTP shows the ability of the tracker to precisely locate the vehicle in the consecutive frames and keeping consistent trajectories. MOTA gives a very intuitive measure of the tracker's performance at keeping accurate trajectories and is independents of its precision in estimating vehicle location.

MOTP represents how the vehicles in the frame are localized.

**Table 5** Test data for tracking

| | view2 | view3 | Total |
|---|---|---|---|
| Number of vehicles | 73 | 107 | 180 |
| car | 40 | 39 | 79 |
| tw | 25 | 51 | 76 |
| Lcv | 4 | 8 | 12 |
| Truck | 3 | 2 | 5 |
| Auto | 1 | 6 | 7 |
| Bus | 0 | 1 | 1 |

$$\text{MOTP} = \frac{\sum d(i,t)}{\sum c(t)}$$

where,

$d\,(i, t)$ = Euclidian distance between vehicle $o_i$ and its detection $h_i$ at time/frame $t$.

$c\,(t)$ = number of matches found for time/frame $t$.

MOTA deals with how many errors like id switching, number of misses, false-positive tracking, false-negative tracking are made

$$\text{MOTA} = 1 - \frac{\sum [m(t)] + fp(t) + mme(t)}{\sum g(t)}$$

where,

$m(t)$ = number of misses at time/frame $t$

$fp(t)$ = number of false-positive at time/frame $t$

$mme(t)$ = number of mismatches at time/frame $t$

$g(t)$ = number of objects present at time/frame $t$

Tables 6 and 7 show the MOTP and MOTA results for the sample tracking data after data imputation.

As can be seen in Table 7, MOTA improved from 94.65% to 98.72%, with the improvement in MOTP from 3.495

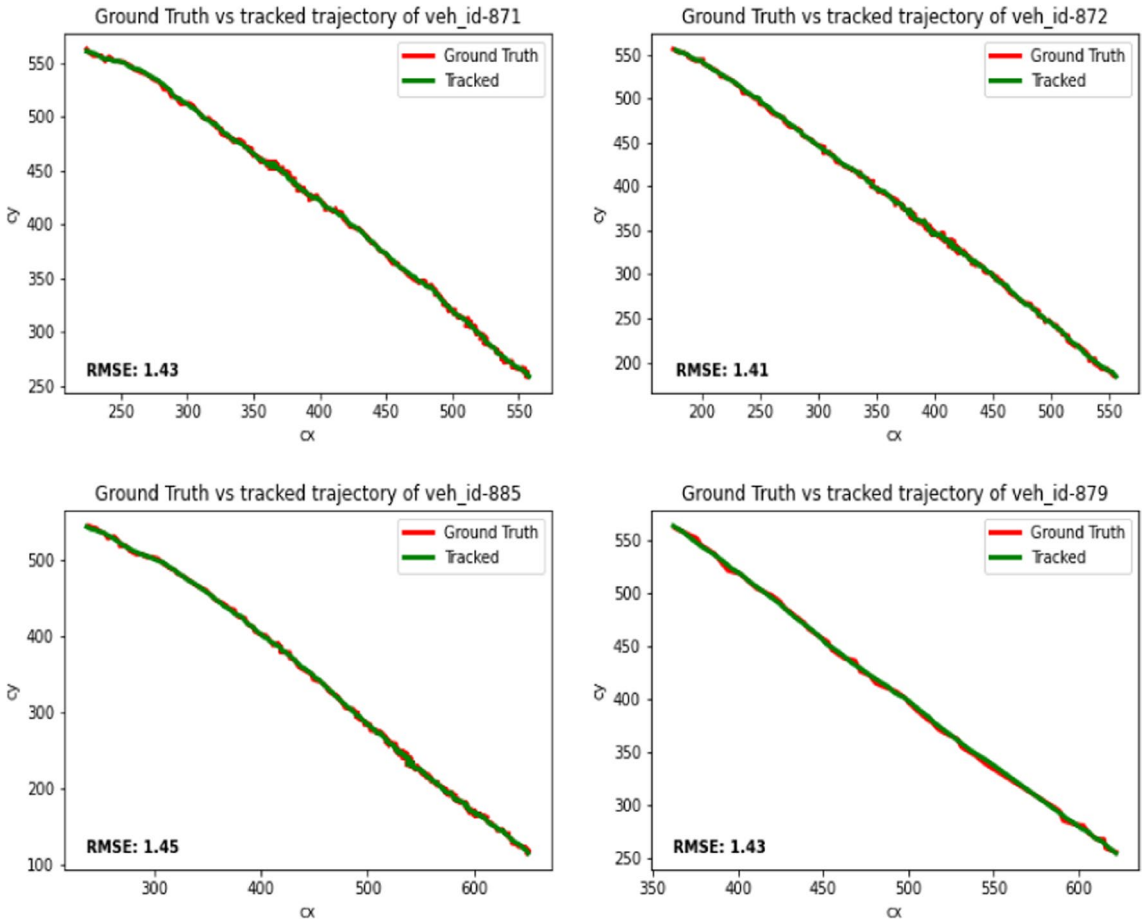



**Fig. 8** Ground truth trajectory vs extracted trajectory comparison

pixels to 3.003 pixels. MOTP for view3 is 3.972 pixels, which signifies that extracted trajectories have a very little deviation, approximately 0.25 m and 0.35 m for view2 and view3 respectively from the actual trajectories. This signifies that the proposed model performed very well for tracking the vehicles with a very small error even in heterogeneous, non-lane-based, and congested traffic conditions.

## Coordinate Transformation

The extracted vehicle positions are in pixel coordinates, separately for view2 and view3. Moreover, the orientation of the road is different in these two views and needs to be aligned before they are transferred to field coordinates to make it a longer road section for traffic flow analysis and modeling. To align the road in both views, the x-axis direction is referred along the median of the road and the y-axis direction is along the perpendicular direction of the median. The axes are transformed in such a way that the road section aligns in both camera views. Figure 10a and b show the pixel coordinates of the four corners of the road section from view2 and view3. Next, to transfer pixel coordinates to the field coordinates, ground truth measurements are taken on the road section: the distance between two consecutive light poles, the width of the road at different sections, the width of white and black markers on the median, etc. To determine the direction perpendicular to the median in a video frame, a 22-feet container truck from a video is taken as a reference. Since the edges of the container are perpendicular to each other, as the truck moves from one end to the other, at different sections truck edges are used as a reference for the determination of y-direction in the video frame. After the determination of the reference axis, world coordinates and their corresponding pixel coordinates are used to get the transformation matrix. Figures 11 and 12 show the pixel coordinates and real-world coordinates used for the transformation matrix.

Since the video is recorded from the top of a building using a regular tripod-mounted camera, a standard pinpoint camera model is used for transformation. This model consists of the matrices which account for all the influencing parameters like Translation of camera, rotation of the camera, scaling, perspective projection, etc. In this case, a real-world reference origin is considered at the bottom of

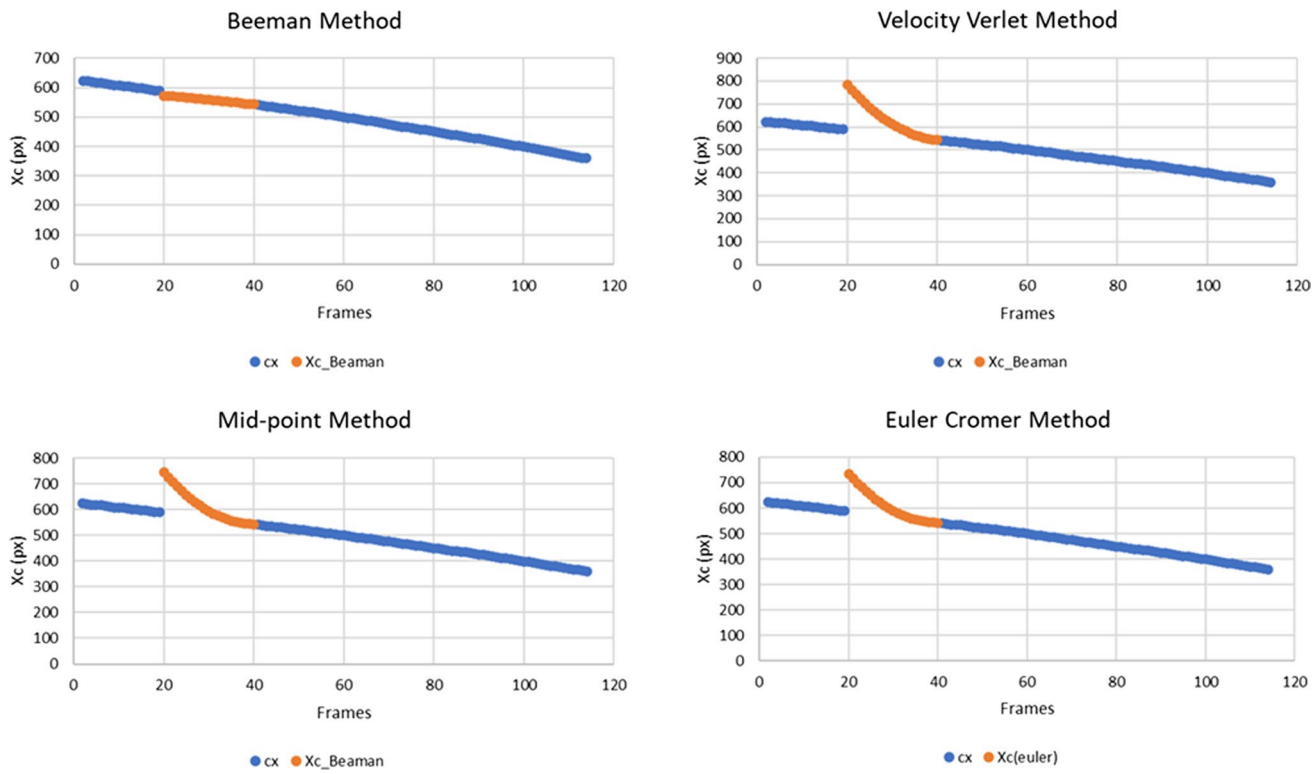



**Fig. 9** Newton integration methods for the imputation of missing positions

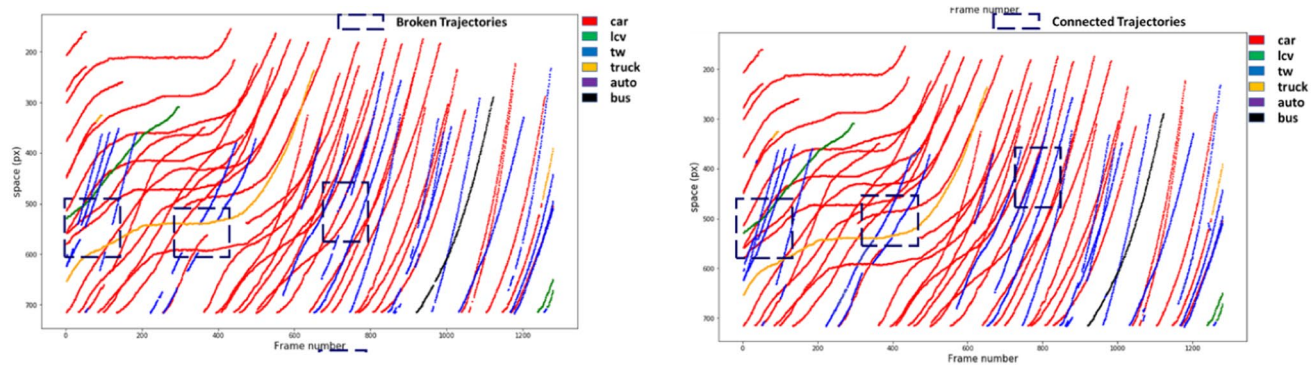



**Fig. 10** Sample results of data imputation

the median. For the camera model following translations are considered.

- Translated the origin to the bottom of the median
- Rotated around Z-axis
- Rotated around X-axis

Then the final transformation is performed using the homographic transformation technique given by the equation:

$$C_h = \mathrm{PR}^{\mathrm{x}}\mathrm{R}^{\mathrm{z}}\mathrm{GW}_h$$

**Table 6** Results of tracking evaluation

| | view2 (before data imputation) | view2 (after data imputation) | view3 |
|---|---|---|---|
| Total number of vehicles present over all frames $\left(\sum g_t\right)$ | 15,527 | 15,903 | 16,684 |
| Total number of misses over all frames $\left(\sum m_t\right)$ | 573 | 203 | 117 |
| Total number of false positives $\left(\sum fp_t\right)$ | 0 | 0 | 196 |
| Total number of mismatches over all frames $\left(\sum mme_t\right)$ | 258 | 0 | 0 |
| Total number of matches found over all frames $\left(\sum c_t\right)$ | 13,124 | 14,057 | 15,540 |
| Summation of the distance between ground truth vehicle location and its corresponding detection location $\left(\sum d_{I,t}\right)$ | 50,065 pixels | 42,217 pixels | 52,119 pixels |

**Table 7** MOTA and MOTP results

| | Ratio of misses (i) $\left[\frac{\sum m(t)}{\sum g(t)}\right]$ | Ratio of false positive (ii) $\left[\frac{\sum fp(t)}{\sum g(t)}\right]$ | Ratio of mismatches (iii) $\left[\frac{\sum mme(t)}{\sum g(t)}\right]$ | MOTA $[1-(i+ii+iii)]\%$ | MOTP $\left[\frac{\sum d(i,t)}{\sum c(t)}\right]$ |
|---|---|---|---|---|---|
| View2 (before data imputation) | 0.0369 | 0 | 0.0166 | 94.65% | 3.495 pixels |
| View2 (after data imputation) | 0.0127 | 0 | 0 | 98.72% | 3.003 pixels |
| View3 | 0.0070 | 0.0117 | 0 | 98.13% | 3.972 pixels |

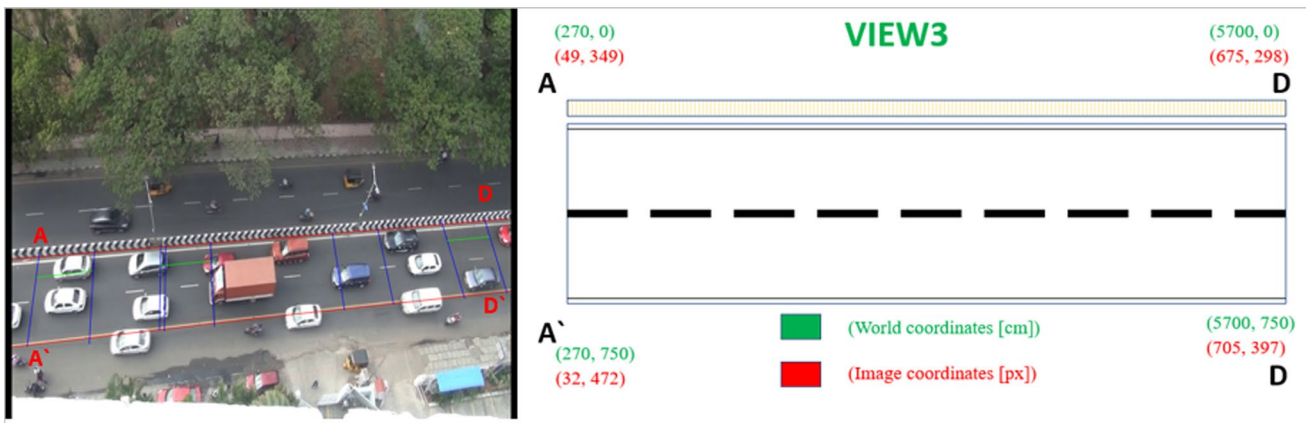



**Fig. 11** View3 reference world axis and coordinates

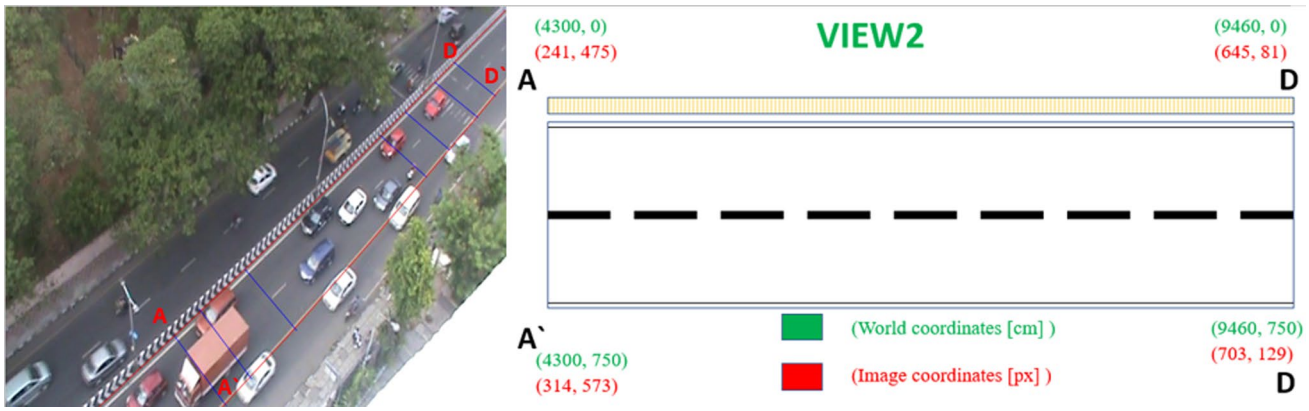



**Fig. 12** View2 reference world axis and coordinates

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

World coordinates — Image coordinates — Transformation Matrix

Where $P$ is the perspective transformation matrix, $R^x$ is the rotation matrix along X-axis, $R^z$ is the rotation matrix along Z-axis, G is the translation matrix, $W_h$ is the world-coordinate matrix and $C_h$ is the homogeneous image coordinates matrix in pixel. Figure 13 shows the trajectories in the field coordinates.

## Discussion

Literature showed that there are very limited studies on trajectory extraction in mixed traffic and congested conditions. To address this gap, this paper proposed a YOLOv3 deep-learning-based framework to extract trajectories from an urban corridor. Two videos recorded from a vantage point near Madhya Kailash intersection in Chennai are chosen for the case study. Both videos have different orientations and scales. The traffic consisted of six-vehicle classes namely car, two-wheeler, bus, truck, three-wheeler, and lcv. A hybrid training dataset was generated from 236 frames (extracted from the two videos). When tested on 38 frames consisting of a total of 832 vehicle instances, the model manages to detect 99.39% of vehicles successfully.

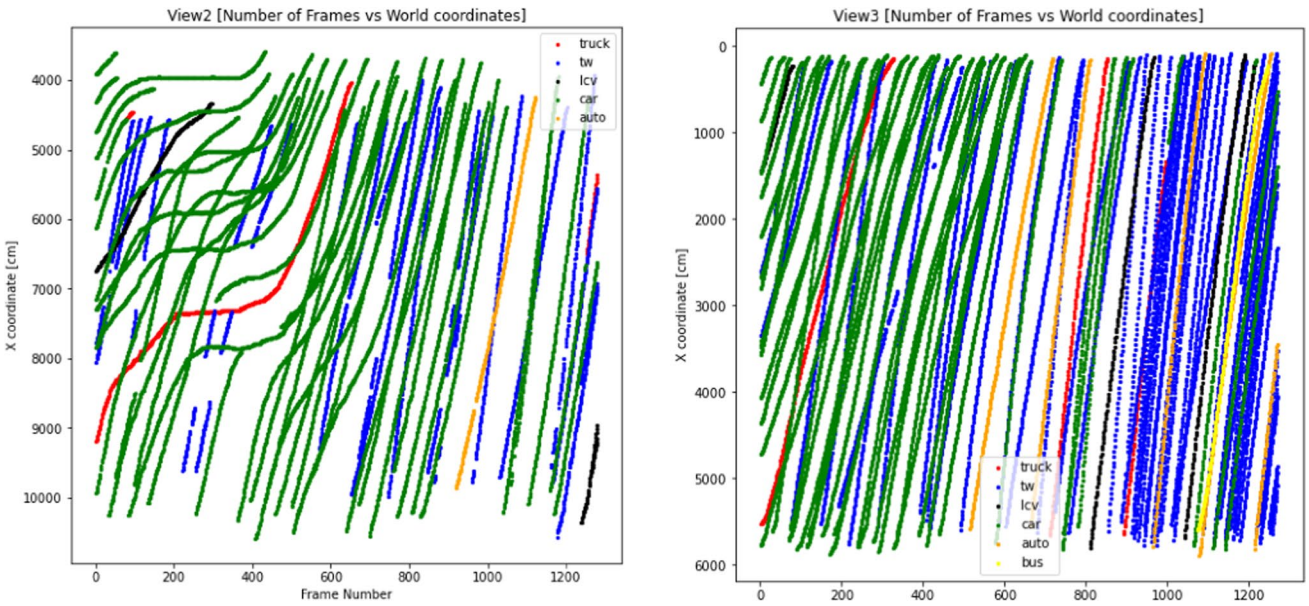



**Fig. 13** Sample trajectories in field coordinates

The performance of the model is evaluated using precision, recall, F1-score, and quality. It was found that cars, two-wheelers, and autos have higher values for all evaluation metrics (greater than 97%) compared to lcvs and trucks. The model performance in terms of object localization is analyzed using the IoU metric. The model showed a mean IoU of 77.54%, which is considered superior in image processing literature.

Detection and classification are followed by tracking and trajectory extraction of multiple vehicles. The deepSORT tracking algorithm, which uses Kalman Filters, is used to track vehicles and extract trajectories. Trajectories extracted from two 50 s video clips from the view2 and view3 are benchmarked with trajectories extracted using the DataFromSky (DFS) platform, which showed that the proposed model performed well for trajectory extraction. There were few cases of missed detection and id-switching due to the roadside trees and occlusion due to large vehicles in view2 whereas, there was one case of false detection in view3. All these errors are identified and corrected in post-processing; broken trajectories are imputed using numerical integration techniques and localized moving average method. The tracking results showed MOTA as 98.72% for view2 and 98.13% for view3 whereas MOTP as 3.003 pixels and 3.972 pixels for view 2 and view3 respectively.

Some of the highlights of this work include:

- Deep Learning Techniques for mixed and congested conditions: The model presented in this paper demonstrated that it performs well under mixed and congested traffic conditions to extract trajectories of a high level of accuracy.
- Hybrid Training data: Hybrid training dataset is generated from two sets of frames of different orientations and scales. It was found that the model trained with the hybrid dataset detected almost all the vehicles in the frame. This can have significant implications for traffic data collection and extraction because any variations in the viewing angles or positions do not require recalibration of the model for video extraction if it is originally trained using sufficiently diverse positions and orientations. This can be extremely useful for drone-based imagery, where imagery from various altitudes does not require different training datasets for each altitude, but one hybrid training dataset.
- Custom anchor boxes: The accuracy and efficiency of the model are improved using customized anchor boxes. The number and sizes of the anchor boxes are determined based on K-means clustering and IoU metric. It is found that custom anchor boxes improve the accuracy and rate of convergence and decrease the computation time.
- Numerical integration methods: Unlike the methods in the literature, this paper proposes the use of numerical integration techniques for trajectory imputation. These methods are based on physical phenomena and better capture the features of the real trajectories.
- Coordinate transformation and error correction methods: The paper also proposed coordinate transformation to align the road segments in two frames and transform the pixel coordinates to real-world coordinates.
- : The paper also proposed a novel error correction method to correct the real-word coordinates.

Based on this experience, vehicle trajectories from 1 h. 50-min videos of both views are extracted and post-processed to make them available for research purposes. It is expected that these trajectories will immensely help the traffic flow community to unveil new phenomena and model driving behaviors in mixed traffic.

**Acknowledgements** The authors would also like to thank Kiran Roy, a former Dual Degree student in the Department of Civil Engineering, IIT Madras for recording the videos, IIT Madras for providing the facilities to conduct the research, and MHRD, Government of India for providing the scholarship to the first author.

**Author contributions** Conceptualization: Bhargava Rama Chilukuri; Methodology: Rohan Dhatbale, Bhargava Rama Chilukuri; Formal analysis and investigation: Rohan Dhatbale, Bhargava Rama Chilukuri; Writing—original draft preparation: Rohan Dhatbale; Writing—review and editing: Rohan Dhatbale, Bhargava Rama Chilukuri; Funding acquisition: Bhargava Rama Chilukuri; Resources: IIT Madras, MHRD (Govt. of India); Supervision: Bhargava Rama Chilukuri.

**Funding** The authors would also like to thank the SPARC program, MHRD, Government of India, and IC&SR, IITM for supporting this research through projects.

## Declarations

**Conflicts of interest** None.

## References

Adu-Gyamfi YO, Asare SK, Sharma A, Titus T (2017) Automated vehicle recognition with deep convolutional neural networks. Transp Res Rec 2645(1):113–122

Apeltauer J, Babinec A, Herman D, Apeltauer T (2015) Automatic vehicle trajectory extraction for traffic analysis from aerial video data. Int Arch Photogramm Remote Sensing Spatial Inform Sci 40(3):9

Barmpounakis EN, Vlahogianni EI, Golias JC (2016) Extracting kinematic characteristics from unmanned aerial vehicles (No. 16–3429)

Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110(3):346–359

Bewley A, Ge Z, Ott L, Ramos F, Upcroft B (2016) Simple online and realtime tracking. In 2016 IEEE International Conference on Image Processing (ICIP) 3464–3468

Bisong E (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. A press, Berkeley https://doi.org/10.1007/978-1-4842-4470-8_7

Chen E, Tang X, Fu B (2018) A modified pedestrian retrieval method based on faster R-CNN with integration of pedestrian detection and re-identification. In: 2018 International conference on audio, language and image processing (ICALIP) pp 63–66

Chen X, Li Z, Yang Y, Qi L, Ke R (2020) High-resolution vehicle trajectory extraction and denoising from aerial videos. IEEE Transactions on Intelligent Transportation Systems

Cheng HY, Yu CC (2012) Detecting and Tracking Vehicles in Airborne Videos. International Journal of Computer and Information Engineering 6(5):665–668

Choi JH, Lee D, Bang H (2011) Tracking an unknown moving target from uav: Extracting and localizing an moving target with vision sensor based on optical flow. In The 5th International Conference on Automation, Robotics and Applications 384–389

Coifman B, Li L (2017) A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset. Transportation Research Part B: Methodological 105:362–377

Danescu R, Oniga F, Nedevschi S, Meinecke MM (2009) Tracking multiple objects using particle filters and digital elevation maps. In 2009 IEEE Intelligent Vehicles Symposium 88–93

DFS (Data From Sky, 2020): https://datafromsky.com/

Gao H, Kong SL, Zhou S, Lv F, Chen Q (2014) Automatic extraction of multi-vehicle trajectory based on traffic videotaping from quadcopter model. Appl Mechan Mater 552:232–239. Trans Tech Publications Ltd.

Garcia F, Cerri P, Broggi A, de la Escalera A, Armingol JM (2012) Data fusion for overtaking vehicle detection based on radar and optical flow. In 2012 IEEE Intelligent Vehicles Symposium 494–499

Gould, H., Tobochnik, J., & Christian, W. (2016). An introduction to computer simulation methods application to physical system.

Guido G, Vitale A, Saccomanno FF, Astarita V, Giofrè V (2014) Vehicle tracking system based on videotaping data. Procedia Soc Behav Sci 111:1123–1132

Haghighat AK, Ravichandra-Mouli V, Chakraborty P, Esfandiari Y, Arabi S, Sharma A (2020) Applications of Deep Learning in Intelligent Transportation Systems. Journal of Big Data Analytics in Transportation 2(2):115–145

He K, Gkioxari G, Dollár P, Girshick RB (2017) "Mask R-CNN", 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988

Henriques JF, Caseiro R, Martins P, Batista J (2014) High-speed tracking with kernelized correlation filters. IEEE Trans Pattern Anal Mach Intell 37(3):583–596

Hue C, Le Cadre JP, Pérez P (2002) Sequential Monte Carlo methods for multiple target tracking and data fusion. IEEE Trans Signal Process 50(2):309–325

Jonathan Hui, "Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)", https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

Jodoin JP, Bilodeau GA, Saunier N (2014) Urban tracker: Multiple object tracking in urban mixed traffic. In IEEE Winter Conference on Applications of Computer Vision 885–892

Kanagaraj V, Asaithambi G, Toledo T, Lee TC (2015) Trajectory data and flow characteristics of mixed traffic. Transp Rese Rec 2491(1):1–11

Khan MA, Ectors W, Bellemans T, Janssens D, Wets G (2017) Unmanned aerial vehicle–based traffic analysis: Methodological framework for automated multivehicle trajectory extraction. Transp Res Rec 2626(1):25–33

Kim Z, Gomes G, Hranac R, Skabardonis A (2005) A machine vision system for generating vehicle trajectories over extended freeway segments. In 12th World Congress on Intelligent Transportation Systems

Kim EJ, Park HC, Ham SW, Kho SY, Kim DK (2019) Extracting vehicle trajectories using unmanned aerial vehicles in congested traffic conditions. J Adv Transp. https://doi.org/10.1155/2019/9060797

Koller D, Weber J, Malik J (1994) Robust multiple car tracking with occlusion reasoning. European Conference on Computer Vision. Springer, Berlin, pp 189–196

Leibe B, Seemann E, Schiele B (2005) Pedestrian detection in crowded scenes. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) 1:878–885

Li Q, Lei B, Yu Y, Hou R (2009) Real-time highway traffic information extraction based on airborne video. In 2009 12th International IEEE Conference on Intelligent Transportation Systems. IEEE. 1–6

Li J, Hajimirsadeghi H, Zaki MH, Mori G, Sayed T (2014) Computer vision techniques to collect helmet-wearing data on cyclists. Transp Res Rec 2468(1):1–10

Miao Q, Wang G, Shi C, Lin X, Ruan Z (2011) A new framework for on-line object tracking based on SURF. Pattern Recogn Lett 32(13):1564–1571

Milan A, Roth S, Schindler K (2013) Continuous energy minimization for multitarget tracking. IEEE Trans Pattern Anal Mach Intell 36(1):58–72

Montanino M, Punzo V (2013) Making NGSIM data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction. Transp Res Rec 2390(1):99–111

Oh J, Min J, Kim M, Cho H (2009) Development of an automatic traffic conflict detection system based on image tracking technology. Transp Res Rec 2129(1):45–54

Ozkurt C, Camci F (2009) Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks. Mathem Comput Appl 14(3):187–196

Redmon, Joseph, et al (2016) "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition

Redmon J, Farhadi A (2018) Yolov3: An incremental improvement. arXiv preprint

Ren S, He K, Girshick R Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Adv Neural Inform Process Syst 91–99

Rodríguez-Canosa GR, Thomas S, Del Cerro J, Barrientos A, MacDonald B (2012) A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera. Remote Sensing 4(4):1090–1111

Rothrock RL, Drummond OE (2000) Performance metrics for multiple-sensor multiple-target tracking. Signal Data Process Small Targets 4048:521–531

Song H, Liang H, Li H, Dai Z, Yun X (2019) Vision-based vehicle detection and counting system using deep learning in highway scenes. European Transport Research Review, 11(1):1–16

St-Aubin P, Saunier N, Miranda-Moreno LF, Ismail K (2013) Use of computer vision data for detailed driver behavior analysis and trajectory interpretation at roundabouts. Transp Res Rec 2389(1):65–77

Tsai LW, Hsieh JW, Fan KC (2007) Vehicle detection using normalized color and edge map. IEEE Trans Image Process 16(3):850–864

Tzutalin (2015) LabelImg. Git code. https://github.com/tzutalin/labelImg

Wang L, Yung NHC (2012) Three-dimensional model-based human detection in crowded scenes. IEEE Trans Intell Transp Syst 13(2):691–703

Wang L, Yung NHC, Xu L (2014) Multiple-human tracking by iterative data association and detection update. IEEE Trans Intell Transp Syst 15(5):1886–1899

Wang L, Chen F, Yin H (2016). Detecting and tracking vehicles in traffic by unmanned aerial vehicles. Automation in construction, 72:294–308

Xu Y, Yu G, Wang Y, Wu X, Ma Y (2017) Car detection from low-altitude UAV imagery with the faster R-CNN. J Adv Transp. https://doi.org/10.1155/2017/2823617

Yilmaz A, Javed O, Shah M (2006) Object tracking: A survey. Acm Comput Surveys (CSUR) 38(4):13-es