Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 63 – 78
© TÜBİTAK
doi:10.3906/elk-2101-93

Motion-aware vehicle detection in driving videos

Mehmet KILIÇARSLAN[*], **Tansu TEMEL**
Department of Computer Engineering, Eskişehir Technical University, Eskişehir, Turkey

Received: 25.02.2021 • **Accepted/Published Online:** 16.06.2021 • **Final Version:** 19.01.2022

Abstract: This paper focuses on vehicle detection based on motion features in driving videos. Long-term motion information can assist in driving scenarios since driving is a complicated and dynamic process. The proposed method is a deep learning based model which processes motion frame image. This image merges both spatial (frame) and temporal (motion) information. Hence, the model jointly detects vehicles and their motion from a single image. The trained model on Toyota Motor Europe Motorway Dataset reaches 83% mean average precision (mAP). Our experiments demonstrate that the proposed method has a higher mAP than a tracking-based model. The proposed method runs real-time in driving videos which enables the model to be used in time-critical applications such as autonomous driving and advanced driving assistance systems.

Key words: Vehicle-motion detection, driving video, object detection, motion profile, spatial-temporal images

1. Introduction

Vehicle detection has been studied over decades for surveillance and dynamic cameras such as autonomous driving and advanced driver assistance systems (ADAS). Dynamically changing background on moving cameras introduces more challenges compared to static cameras. Shape-based vehicle detection systems in either static or dynamic cameras provide only the vehicle position in a frame. For safe driving, both vehicle position and direction are crucial components of a detection system. A vehicle's moving direction is necessary for understanding the vehicle-vehicle interaction. Alternative to other methods that recognize vehicles in each video frame and then track them across video frames, we start from motion analysis and directly extract the vehicle motion. A target vehicle in the environment is within the visible scope for detection, while an ego vehicle carries the camera system.

Relative image motion between a target vehicle and an ego vehicle involves common properties. These properties are much more different from a static background and other dynamic objects in the environment. We can profile the driving environment into the spatial-temporal motion profile images (MPI) [1, 2]. These images are generated by averaging a region around the horizon to capture horizontal motion similar to Figure 1b. In this work, we use motion profile patch images (MPPI) which are temporal patches from MPI instead of the whole MPI. All the objects have continuous and smooth motion in the MPPI because of the limited ego vehicle maneuver capabilities. However, one cannot conclude any given two pairs of traces on an object as a vehicle from motion trajectories only. Hence, we further construct the motion frame image (MFI). This image concatenates MPPI with a video frame to capture both vehicle shape and motion on a single image as displayed in Figure 1c. We thus used existing conventional shape-based object detection methods on a newly introduced

[*]Correspondence: mkilicarslan@eskisehir.edu.tr

MFI to identify both a vehicle and its motion direction. It turns coupled spatial-temporal feature extraction problems into a unified single image processing, as in the traditional shape-based object detection methods.

Our detection paradigm classifies target vehicles based on their relative motion and separates them from the dynamic background. Our goal is to classify vehicles based on their motion direction. For this goal, we have adapted the state-of-the-art deep learning shape-based object detection architecture YOLOv3 [3] to process MFI and achieved good accuracy in real-time.

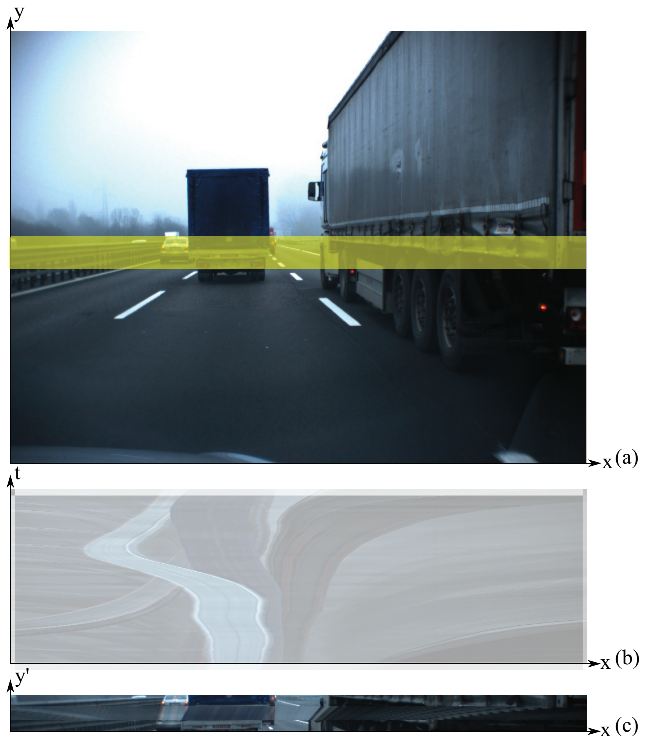


Figure 1. (a) The last frame from a 52 s long video sequence. The horizontal yellow belt is the vertical averaging region for the motion profile image. (b) Motion profile image generated out of a video sequence for visualization. (c) Merging video frames and motion profile patch image ($MPPI$) results in motion frame image (MFI) for real-time processing. The horizontal axis is the same as the video frame, while the vertical axis, y', is a mixture of spatial and temporal axes.

As a summary of the paper, Figure 2 gives a diagram of processing from motion acquisition to vehicle classification based on motion information. The main contribution different from others is the vehicle and motion classification in the MFI detailed as follows: (1) We introduce a unified spatial-temporal image, MFI, to sense both shape cues from a frame and temporal information from MPPI. (2) We procedurally assign vehicle classes based on their relative motion for training. (3) We condense multiple video frames into a single reduced dimensional image. Hence, the deep learning model can process more efficiently both in training and testing time. (4) The proposed method is tested on publicly available Toyota Motor Europe Motorway Dataset (TME) [4] and reached 83% mean average precision. Further significance is to detect a vehicle and its motion efficiently in a driving video. The method uses the minimum amount of data that requires no other postprocessing stages as vehicle tracking.

The primary input source is video sequences in our processing pipeline. We transform these sequences into spatial-temporal images. Vehicle detection methods in the literature are image-based, while vehicle motion detection methods use prior vehicle bounding boxes over consecutive frames in a video sequence. Therefore, we will go through image-based vehicle detection methods and then video sequence-based vehicle motion methods later.

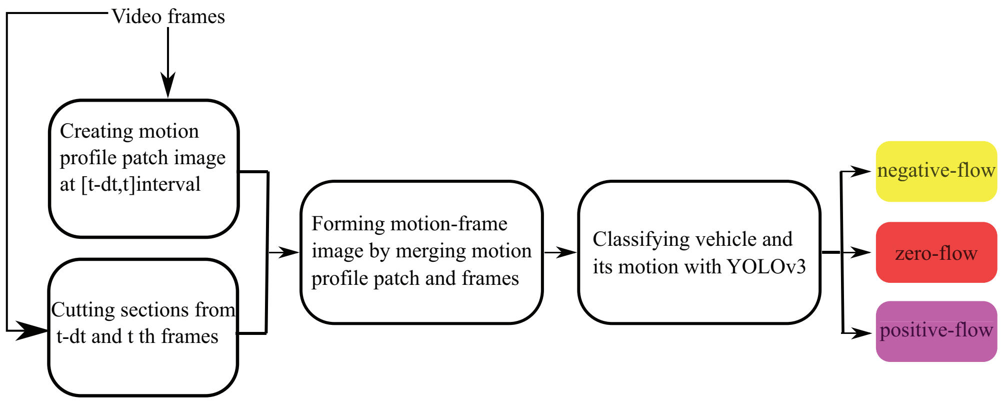

Video frames
Creating motion profile patch image at [t-dt,t]interval
Cutting sections from t-dt and t th frames
Forming motion-frame image by merging motion profile patch and frames
Classifying vehicle and its motion with YOLOv3
negative-flow
zero-flow
positive-flow

Figure 2. A diagram to classify vehicles according to trajectories in the motion frame images.

Researchers have studied vehicle detection in still images for many years and achieved highly accurate results. The key features are the appearance of vehicles that have large variations in shape, size, orientation, and occlusions. In early studies, these features are shadows [5–7], edges [8], corners [9], symmetry [10, 11], texture [12], and color [13]. These preliminary features transform into more general and robust feature sets. These additional features allow direct vehicle classification and vehicle detection in still images. Histogram of oriented gradients (HOG) [14], Haar-type feature [15], local binary patterns (LBP) [16] are the early hand-crafted feature extraction methods in classification. A sliding window (bounding box) over an entire frame captures features for detecting a vehicle. The accuracy of these methods is at a moderate level.

More recently, the deep learning based convolutional neural network (CNN) methods replaced hand-crafted features. These methods have achieved higher accuracy by learning vehicle and background variations from image datasets. The trained model can memorize shape and structure in a deep architecture with nodes, layers, and weights [3, 17, 18]. A deep learning model can learn the vehicle structure under normal illumination, although the model requires additional data for night driving and adverse weather scenarios. One reason for CNN's success is publicly available large amounts of datasets. YOLOv3 [3], and several new approaches such as fast R-CNN [19], and SSD [20] have selected anchor boxes based on the frequency of target occurrences, which narrows down the search of target bounding boxes and speeds up recognition drastically.

Video-based methods can use image-based vehicle detection methods sequentially in each video frame. However, detection algorithms use sparse key-frames for efficiency. Then, another tracking module merges these detections. Most researchers use optical-flow based approaches to extract motion features [21]. Many works coupled optical flow with tracking for vehicle classification [22, 23]. Deep-sort [24] can detect and track objects in real-time. The common procedure of all these works is the additional required tracking module. Besides these, [25] models motion information through tracking the corner features for vehicle detection with hidden Markow Model without using the optical flow. Also, [2] uses both long-term horizontal and vertical image motion for time-to-collision computation through motion profile images. However, the method cannot distinguish vehicles from other objects.

Different vehicle actions create unique motion footprints in driving scenes. A vehicle must be detected first to understand vehicle-vehicle interaction. However, this generates a high dependency on the vehicle detection system. The framework in [26] classifies vehicles into various categories by first detecting vehicles with YOLOv3, then extracts motion with another CNN. As a general video action prediction model, long short-term memory (LSTM) is used for target vehicle trajectory prediction [27]. However, LSTM models require more hardware resources than traditional CNNs, which may degrade the real-time performance.

This paper reformulates vehicle and its motion direction detection into a unified framework. We detect all at once without tracking each video frame in real-time. The primary aim is to classify vehicle motion direction in the spatial-temporal domain by applying highly accurate deep learning methods. Therefore, both spatial shape and temporal motion features in the MFI are the primary source to recognize objects.

In the following, the paper continues with target vehicle motion sensing in driving videos. We detail the proposed method in Section 3. Section 4 discusses the experiments and evaluation results, followed by the conclusion in Section 5.

2. Target vehicle motion in driving videos

2.1. Motion profile images

To grasp horizontal motion in a video, we use spatial-temporal images called motion profile images (MPI) similar to [1, 2]. A 3D video is condensed into an MPI which reveals the relative motion of objects. This image turns spatial and temporal features into shape features like a motion trajectory. A front-facing camera near the back mirror of a vehicle captures the environment. We locate a fixed height horizontal belt near the projected horizon in a frame. This belt covers the entire horizontal field of view including far, and near vehicles, as shown in Figure 1.

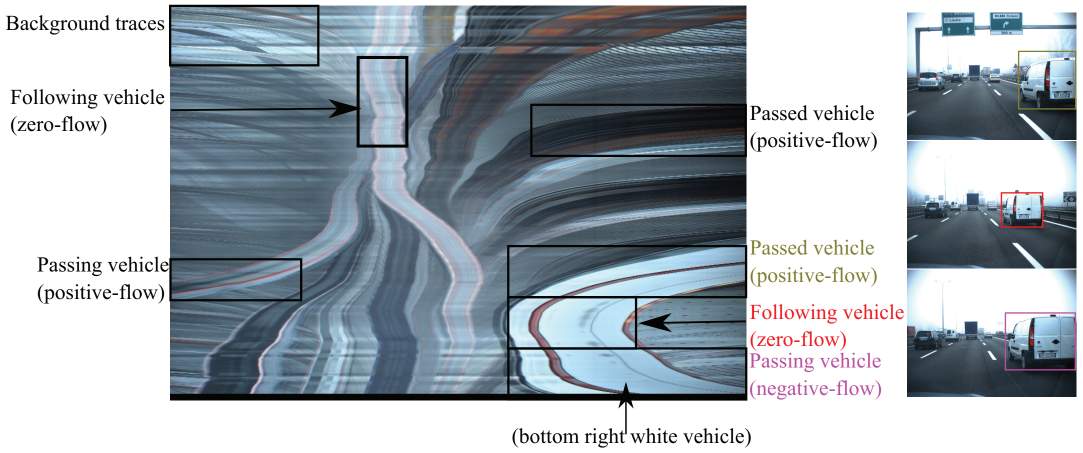

Background traces
Following vehicle (zero-flow)
Passing vehicle (positive-flow)
Passed vehicle (positive-flow)
Passed vehicle (positive-flow)
Following vehicle (zero-flow)
Passing vehicle (negative-flow)
(bottom right white vehicle)

Figure 3. Different motion trajectories in the motion profile image. In the bottom right corner of the motion profile image, the white vehicle passes three states along its whole trajectory. The corresponding video frames are visualized in time order.

We vertically average pixel values to form a 1D array for each frame. This array in consecutive video frames is stacked to generate the motion profile image, $MPI(x,t)$.

$$MPI(x,t) = \sum_{y=h}^{h+dh} I(x,y,t) \tag{1}$$

where h is at the horizon, and dh is the belt height. All vertical objects, like vehicles, leave their motion traces on this image. Its horizontal axis is the same as the video frames, while the vertical axis is the time up-wards. The traces of the objects show longer-term motion than optical-flow. Averaging within a dh range guarantees smooth and continuous motion of traces in driving videos. Vehicle pitching and yawing result in a discontinuity in brightness. Figure 3 shows this effect as horizontal stripes as intensity change across time. Our method does not require brightness consistency like optical-flow based methods. Hence, discontinuity in the intensity brightness will not affect the motion understanding. One MPI captures all the vehicles in the entire horizontal field in an ideal flat road. However, the belt region may miss far vehicles because of the horizon change in a nonflat road. Using multiple MPI like in[2] will overcome this issue. A motion profile patch image (MPPI) from $\mathrm{MPI}(x, t-dt)$ to $\mathrm{MPI}(x,t)$ is used in the vehicle detection pipeline for real-time processing where t is the current frame and dt is the number of previous frames.

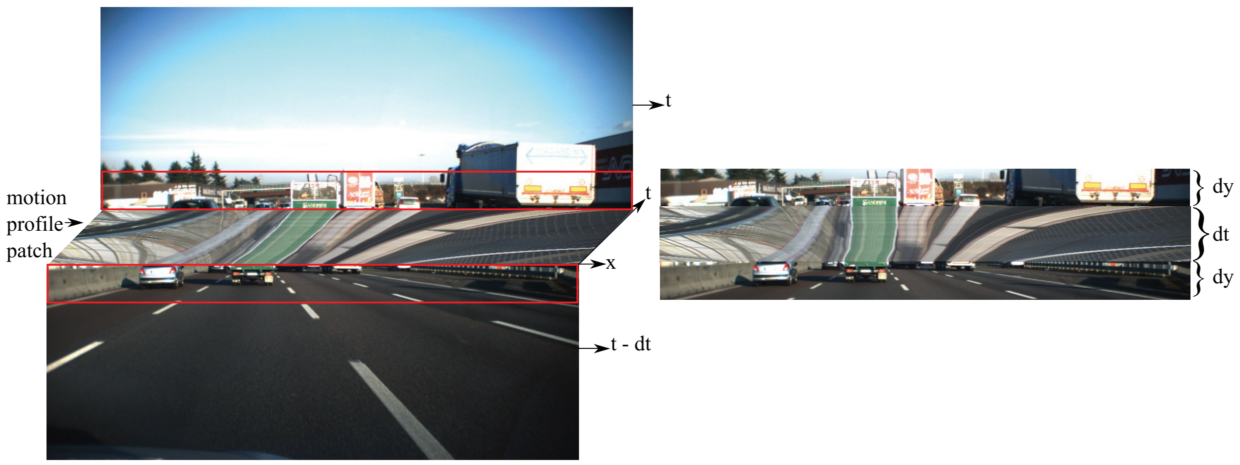


Figure 4. Construction of motion profile patch image (MPPI) Left: MPPI in $[t-dt, t]$ range and entire frames are shown. Right: Generated motion frame image. Instead of using the entire frame, only the dy ranges from frames are used in the final image.

2.2. Motion model of target vehicles

The MPI inherently encapsulates dynamic objects and static background's horizontal image motion. Longer-term motion trajectories can be observed either for a postprocessing analysis or real-time processing if only previous time sequences are used as a motion profile patch image. We use only these patches in our detection system, not the entire MPI sequence.

We generate these motion profiles on driving video datasets. By analyzing these images, different vehicle motions create unique and distinguishable traces. Figure 3 shows various motion patterns of surrounding vehicles and background traces, which we will briefly explain in the following.

- Background trajectory expands from the focus of expansion (FOE) to the sides. Depending on the ego vehicle speed, motion trajectories slope vary.
- Following vehicle trajectory has small horizontal image velocity. It is corresponding to near vertical traces in MPI.
- Passing vehicle trajectory expands from center to either side depending on the right or left passing cases.

- Passed by vehicle trajectory starts from sides to the FOE. The slope depends on the relative speed of the target and ego vehicle.

- Oncoming vehicle trajectory is like a background trajectory with increased relative motion.

Analyzing these motion characteristics, we can observe vehicle motion in the MPI, which gives both its location and the relative motion in almost every depth. A vehicle's position in a single video frame is not enough to understand its direction. The white vehicle at the bottom right in Figure 3 is roughly at the same position, but it has three different actions in its entire journey: passing, following, passed by. Hence, horizontal motion is necessary to understand target vehicle action.

2.3. Merging motion profile patch images with video frames

Target vehicle's shape features are observable in MPI as two tail light traces, its width, symmetry of it, etc. The MPI captures the relative motion of a target vehicle as well. However, it may be ambiguous to spot a trace as a vehicle in the motion profile patch image (MPPI) due to temporal locality. In a highly complex background, far away background objects may have similar trajectories to the target vehicles. Similarly, space in between two vehicles causes ambiguities. Hence, an MPPI alone is insufficient for vehicle detection.

To overcome these ambiguities, we not only rely on traces in the MPPI but also use video frames by merging MPPI with frames. We construct the motion frame image (MFI) which is a concatenation of motion profile patch image in $[t-dt, t]$ interval, sections of frames at $t-dt$ and t. The MFI only uses the previous information up to the current frame. We preserve the shape continuation and smoothness by using only the higher and lower sections near the horizon at $t-dt$ and t frames, respectively. The MFI is defined as

$$\begin{aligned} MFI(x,y,t) = & I(x, h-dy : h, t) \oplus \\ & MPI(x, t-dt : t) \oplus \\ & I(x, h : h+dy, t-dt), \end{aligned} \tag{2}$$

where dy is the spatial interval taken from frames and dt is the patch size, and $\oplus$ is the vertical concatenation operator. Figure 4 shows an MFI example with a subsection of frames and motion profile patch image.

MFI turns the spatial-temporal features into a shape feature extraction problem. These shape-based feature extraction problems have been studied for decades and have achieved highly accurate results. Hence, the vehicle classification based on its motion in MFI is suitable for any shape-based object detector.

3. Methods

3.1. Motion modeling for training

A correctly labeled dataset is crucial to create a successful deep learning model. There has been no labeled dataset based on vehicle motion trajectories so far. Labeling a dataset in the MPPI based on the motion is an ambiguous task for an annotator. Therefore, we manually correct all frame based on bounding boxes. Then use these bounding boxes in the ground truth motion computation. We identify three primary vehicle motion directions. These directions are critical for both autonomous driving and advanced driving assistance systems.

Based on the motion model of target vehicles; we aim to classify motions into:

$$\begin{aligned} &Negative-flow: -\theta_1 < \theta < -\theta_0 \\ &Zero-flow: \quad -\theta_0 \leq \theta \leq \theta_0 \\ &Positive-flow: \quad \theta_1 > \theta > \theta_0 \end{aligned} \tag{3}$$

where θ is the relative horizontal motion between an ego vehicle and a target vehicle. We compute the motion from vehicle's ground truth bounding boxes at $(t-dt)$ and tth frames by the following formula:

$$\theta = atan(\frac{x_c(t) - x_c(t-dt)}{dt}), \tag{4}$$

where x_c is the horizontal center of a bounding box and dt is the time interval. This gives a range of values in $-90° < \theta < 90°$. Figure 5 visualizes the long-term motion of target vehicles.

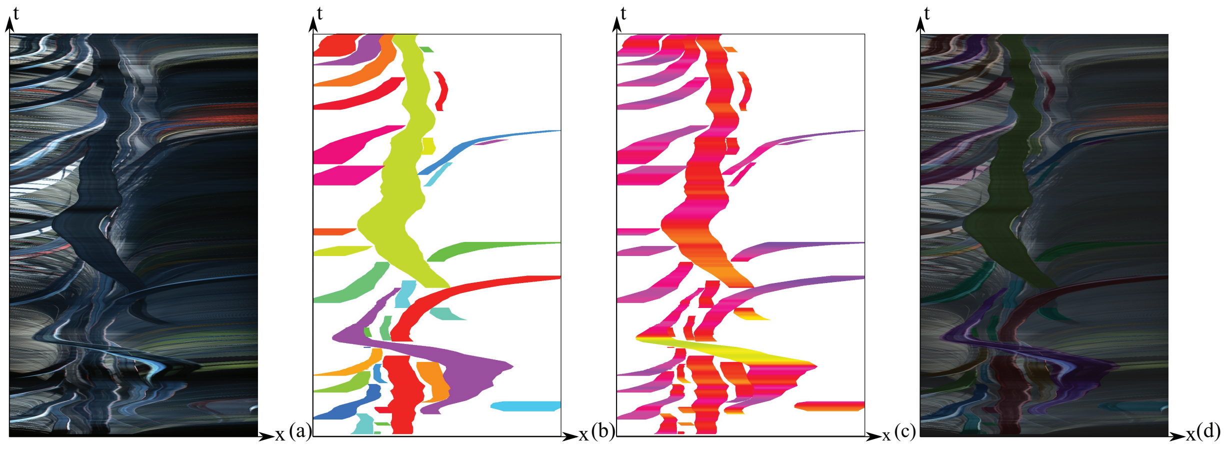


Figure 5. The long term vehicles and their motion visualization in the motion profile image. (a) A full sequence motion profile image. (b) Tracked vehicle bounding boxes across frames projected on the spatial-temporal domain. (c) Motion directions of vehicles. (d) Vehicle tracks are merged with the motion profile image.

These motion classes relate directly to the target vehicle motion model described in Section 2.2. The following vehicle trajectory is the zero-flow regardless of its position. A passing vehicle trajectory is positive-flow where the vehicle is on the right side. A passed by vehicle trajectory is positive-flow where the vehicle is on the left side classes.

3.2. Deep learning architecture

In recent years, object detection based on convolutional neural network (CNN) methods has gained attention. These methods provide both accurate and fast detection results. One of the effiecient and accurate methods is you only look once (YOLO). It is one of the one-stage based methods. For these reasons, we use its variant in this work with a Darknet-53 as a backbone.

Anchors: In YOLOv3, anchors play an important role in object detection accuracy. Anchors are precomputed or assumed bounding box shapes of a target object. Although these anchors depend on the data, there is a direct relationship between the anchors and physical principles, especially in the vehicle detection context.

There are two main factors in determining the size and shape of anchors. One is the aspect ratio of the objects. The anchors should be square, horizontal, and vertical shaped rectangles in conventional YOLOv3. The other factor is the size of the anchors. It directly relates the depth of the vehicles to the anchors.

In MFI the bounding boxes are transformed from frame bounding boxes by

$$f(x, y, w, h) = (x, y_c, w, dt), \tag{5}$$

where x, y, w, h are bounding box's horizontal center, vertical center, width, and height, respectively in a frame, y_c is the center of the MFI, and dt is the patch size. Hence, different from the conventional YOLOv3 anchors, there is no dependency between the width, w, (spatial) and height, dt, (temporal) of the bounding boxes in the MFI more specifically in the motion profile patch images. There is no need to cover the distinct size aspect ratio of anchors. Therefore, anchors must be only in different sizes to cover various target vehicle scales: close, mid, and far distance vehicles.

Dimensions of training images: YOLOv3 architecture expects input image dimensions to be multiples of 32. It is due to five level convolution layers with strides of 2 pixels. Each of these layers results in half the size of its input. Hence the height of 32 pixels ends up in a pixel height. To be consistent with YOLOv3, we keep the height of MFI to $2dy + dt = 32k$ where k is a positive integer.

3.3. Motion frame image formation

There are different configuration options when constructing the MFI from (2) which depends on patch size, dt, and spatial interval, dy. The chosen parameters affect the deep learning model training stage despite their independence.

1. Choosing *dt* much more smaller than *dy*: In this case, the shape features from frame sections will have more regions than the motion profile patch images in the MFI. Since the classes are determined based on motion directions, the deep learning model may not learn the temporal features in MFI.

2. Large *dt* or very small *dt*: Independent of dy, choosing the right size of the temporal resolution, dt, is very significant. Tiny dt may not show any significant temporal action footprint. Also, in the training stage, the information will be lost in the early pooling layers. On the other hand, a wide dt may cover more than one action in its duration. Hence, this will impose difficulty on the training stage due to increased motion variations.

We set dt to 12 frames (0.6 s) in MFI as a motion profile patch image by taking these considerations. However, we resized these patches into 24 pixels to be coherent with YOLOv3 pooling/downsampling layers. Similarly, we set dy to 20 pixels. As a result, MFI resolution turns into 64×1024 pixels which are ~ 12 times smaller than a regular frame yet capture both shape and action of a target vehicle.

Although the MFI consists of both frames and motion profile patch image, the vehicle bounding boxes are drawn on motion profile patch regions. The bounding box width in the tth frame is used as a width in the MFI. Samples from our training images are visualized in Figure 6.

4. Experiment and results

4.1. Dataset and relabeling

There are many publicly available driving video datasets. We consider datasets based on their frame rate, labeling rates, bounding box association across frames, and entire video sequence availability. KiTTI [28] is one

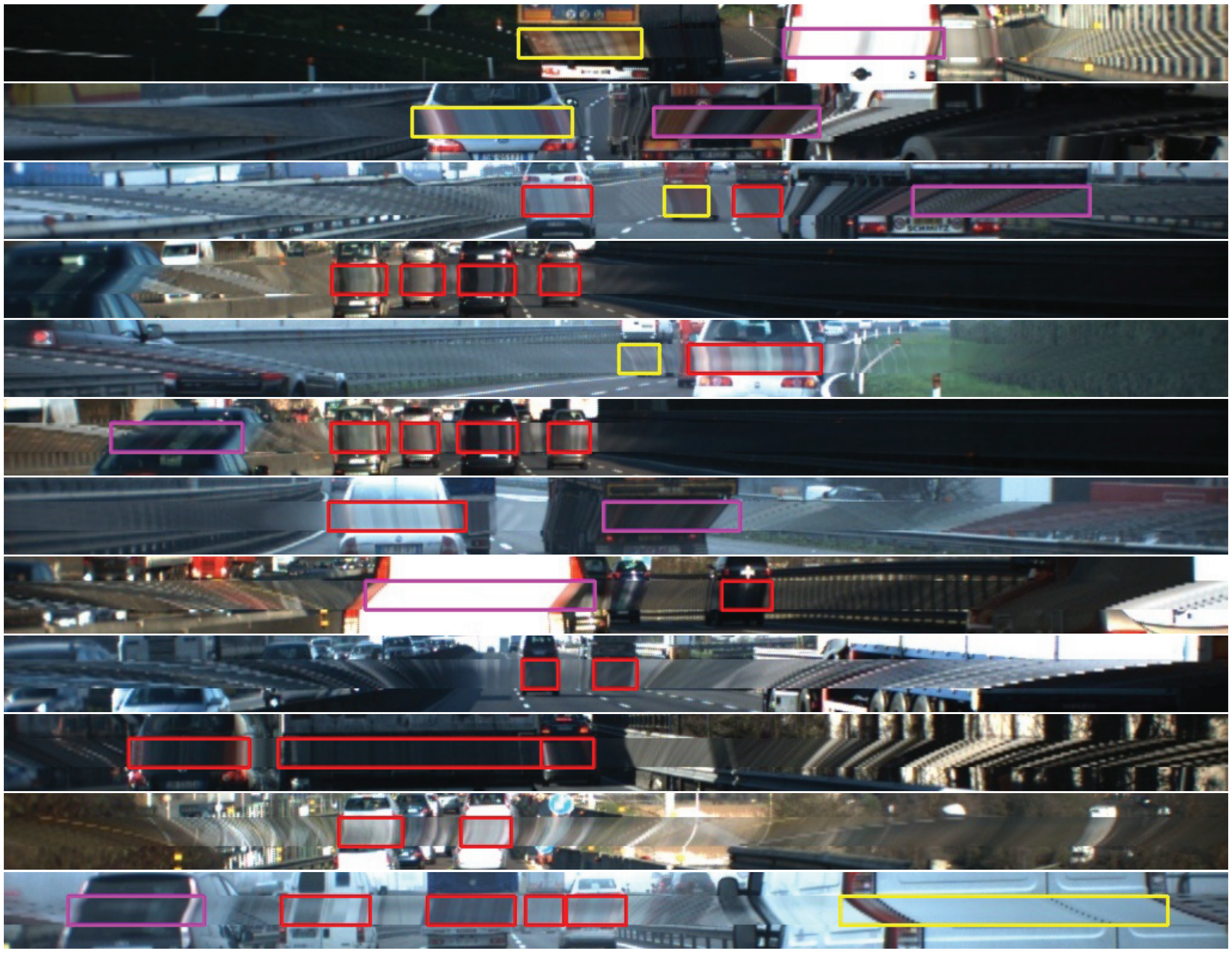
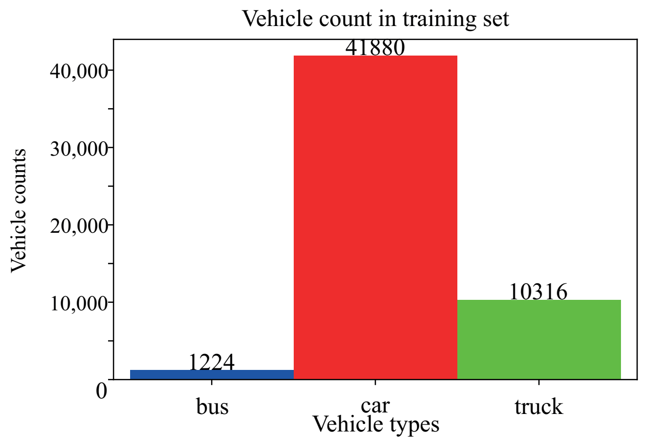


Figure 6. Vehicle labeling in the motion frame images. Bounding boxes are only drawn on the motion profile patch images. The color shows vehicle's motion directions; red: zero-flow, yellow: negative-flow, purple: positive-flow.

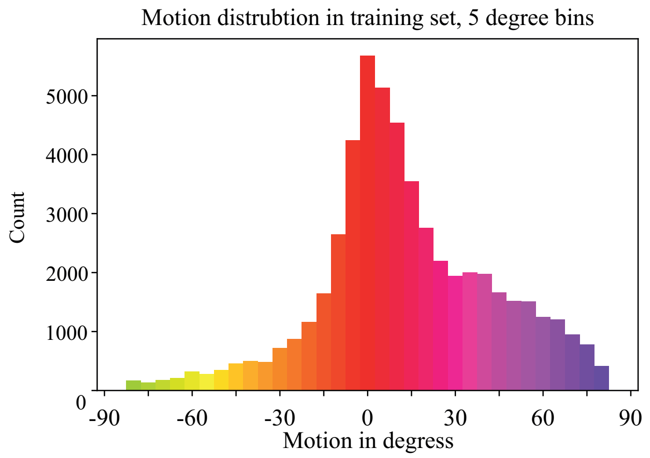
of the widely used datasets. However, it is low in temporal resolution and only provides two frame associations across frames. One of the largest datasets, Cityscapes [29], provides only sparse labeling of objects. Unlike the most publicly available datasets, we use another publicly available TME dataset [4]. It has vehicle association across frames (tracked vehicle labeling) in every frame, provides all the driving videos with a moderately high frame rate. We use this dataset for motion profile generation, motion-frame image construction, training the deep learning models, and evaluating the trained models.

In the TME dataset, there are 28 video sequences under different illumination, traffic scenarios in highway driving environments. We use the daytime videos, which are in total 18 videos. The video resolution is 768 by 1024, and the frame rate is 20 Hz which is sufficient when we turn the video frames into MFI. However, a higher temporal resolution would cause higher quality images and eliminates temporal aliasing. We split the videos into training, testing, and validation sets according to 70%, 20%, and 10%, respectively.

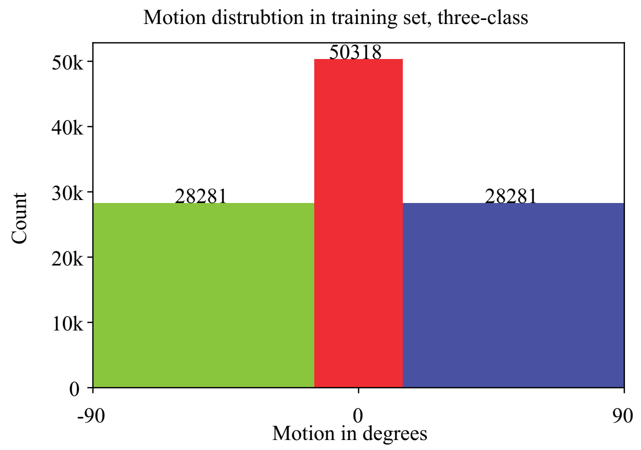
Unfortunately, the TME dataset provides radar-based semiautomatically labeled bounding boxes. These bounding boxes are off from vehicles. A precise bounding box is necessary to compute the ground truth motion



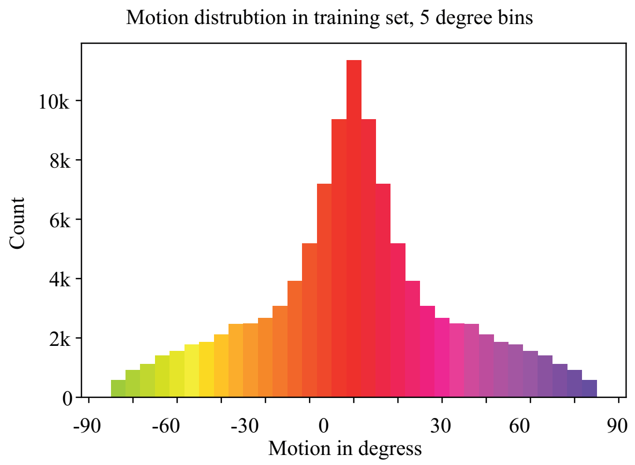
(a) Number of occurrences of different vehicle types.



(b) Motion distribution in the training set. The data is skewed in the positive motion.



(c) Dataset distribution in the training set for three class motion classification.



(d) Motion distribution after the data augmentation by vertical flipping.

Figure 7. Distributions of the dataset in terms of (a) vehicle type counts, (b) motion directions before augmentation, (c) three class motion counts, and (d) motion directions after data augmentation. Color represents motion directions in (b) and (d).

accurately with (4). Hence, we adjust all the bounding boxes to cover only the rear side of vehicles in the original frames. Occluded and truncated vehicles are handled with special care. An occluded vehicle causes an inaccurate vehicle center location in the ground truth data. As a result, the computed ground-truth motion will be wrong. Therefore, whenever a vehicle is partially occluded, we strictly label it as an occluded vehicle. Similarly, for the same reason, a vehicle seen from the side is only labeled whenever its rear side is visible. Also, since the videos are recorded on the highway, all oncoming vehicles are occluded by road guardrails. Therefore, we exclude all of them from the training set. Figure 7a shows the total vehicle count in the training dataset.

The ground truth motion distribution in the training set is shown in Figure 7b. As can be seen from the figure, the motion distribution is highly skewed on the positive flow. However, this type of distribution is expected in left-side driving countries. The majority of vehicle-vehicle interactions generate positive flow. The most obvious cases are: passing a vehicle through on the left side and passed by a vehicle on the right side.

The θ_0 and θ_1 parameters in (3) are set to 15° and 85°, respectively for determining zero, positive, and negative flow classes. Due to the broad coverage of motion intervals, all slow and fast moving vehicles are covered. Also, we set the belt location y to and its height dh to 370 and 30 pixels, respectively. The location of the y is near the horizon, and it depends on the camera recording. The dh is the averaging height, and any vehicle height greater than $dh/2$ will have motion trajectories in the motion profile patch image. Since all the videos are recorded from the same camera, we use fixed y and dh for all videos. Otherwise, different video sequences require different values, which depend on their horizon location.

4.2. Data augmentation

Data augmentation is an approach to replicate the data by various transformations to enlarge the training set. The ground truth motion is oversampled in the positive relative motion that results in a class imbalance. We augment the data by vertically flipping every training image to overcome the imbalanced class problem. Positive and negative flow samples equate after the augmentation. After flipping the images, the motion distribution is weighted equally on positive and negative flows, as shown in Figure 7d. Figure 7c shows the number of samples in each of the zero, negative and positive flow classes after the augmentation.

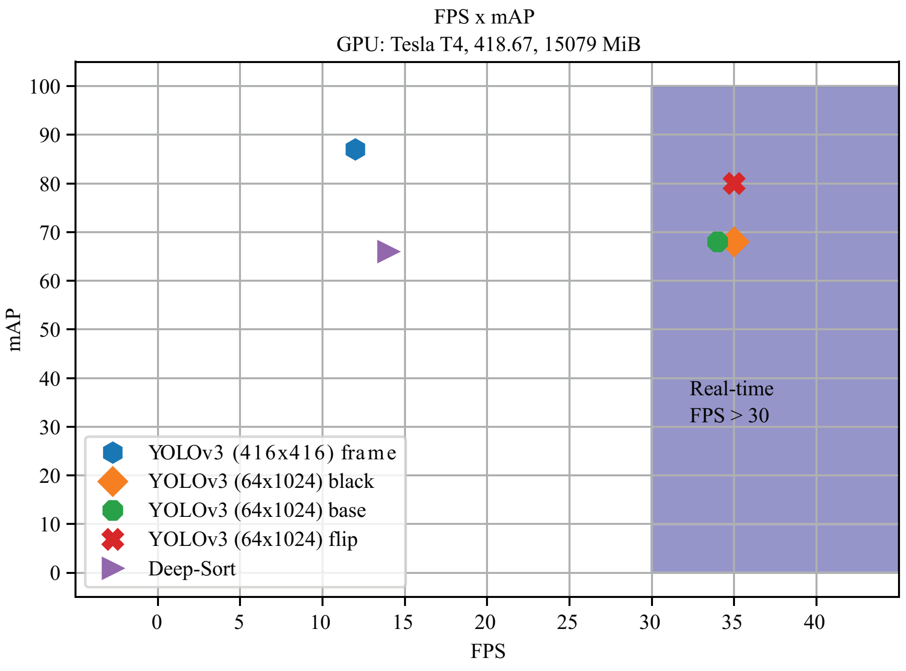


Figure 8. The frame per second (FPS) and mean average precision plot of the conducted experiments.

4.3. Training and evaluation results

We have conducted five sets of experiments on an Ubuntu 18.04 machine with Intel(R) Xeon(R) CPU @ 2.20 GHz and Tesla T4 GPU hardware. For all the training experiments, the default YOLOv3 parameters are used. The base, black, flip, and frame experiments are conducted using the Darknet framework while the pytorch framework is used in the deep-sort based tracking method. Different from the default settings, in the base, black, and flip experiments anchors are set to $[24 \times 40], [24 \times 120], [24 \times 350]$ to cover close, middle, and far distance range vehicles. In all experiments, a detection is counted as a true detection if the intersection over union (IOU) of detection and the ground truth bounding box is greater than 0.5. Furthermore, we use only the training data for training each model and the testing data for computing evaluation metrics. Overall comparison in terms of both mean average precision and running time in frame per second rate are plotted in Figure 8. Table shows the mean average precision evaluation results and configuration parameters for all experiments. We also compute mean average precision metrics for the training set. The training set mAP values are all around 99% except for the deep-sort method. The smaller size of MFI, (64×1024), compared to the frame size resolution (416×416) in training results in higher fps of the proposed method. Figure 9 highlights bounding box locations in the conducted experiments. Result videos can be accessed from the following link: https://drive.google.com/drive/folders/1-orf8Y8ysbnweX13MXUkPcQwI2r5BJRi?usp=sharing

Training on frames (frame): The training sample size is an important property for success in any deep learning based training model. Training the TME dataset with video frames helps us understanding the dataset capacity to compare it with traditional models. Also, we use these training weights in deep-sort [24] (detect-track based) method as a baseline comparison. Using a pre-trained network from other datasets for the deep-sort model imposes some problems. The ground truth data in our labeling only covers rearview vehicles. However, a pretrained network from other datasets will include both oncoming and rearview vehicles. Hence, all the oncoming vehicles will be considered as a false positive, and the accuracy will be largely degraded. The frame based training model achieved 90% of average precision for the vehicle detection task. Additionally, using YOLOv3's network weights pretrained on ImageNet dataset for transfer learning slightly increased the mAP to 93%

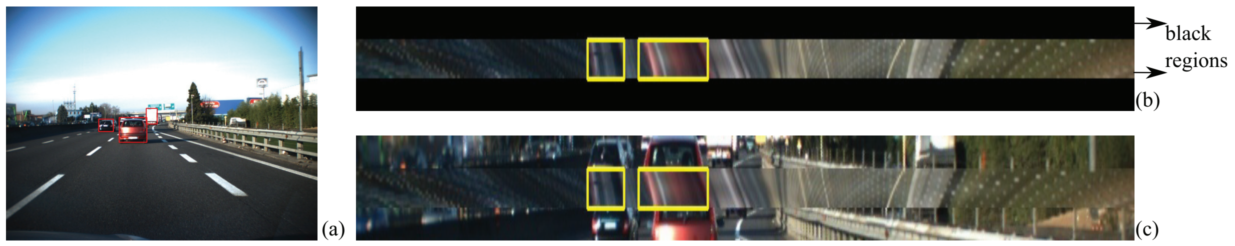


Figure 9. (a) Bounding boxes in video frames. (b) Bounding boxes in the MFI where top and bottom regions are set to black color. (c) Bounding boxes in the MFI. Bounding box colors in (b) and (c) represent motion class. (b) and (c) are enlarged for better visibility. Vehicles without bounding boxes are occluded.

Using black pixels on MFI (black): This experiment's main objective is to understand the importance of the frame sections in MFI. In this experiment, MFI is constructed with only motion profile patch images, and the rest of the pixels coming from the frames are set to black color. The training set for this experiment also includes flipped images. Figure 9b shows a training sample image with vehicle bounding boxes where black regions are on top and bottom of the image. The training results showed a low ability to learn three motion classes from only vehicle traces. One reason is the motion ambiguity of intraclass vehicle motion similarity and similar traces from other objects from the background. The mean average precision in the testing set is around 73%, and average precision is $71\%, 74\%$, and 75% for zero, negative, and positive flows, respectively.

Training on MFI (base): This configuration is the first set of experiments which uses the proposed MFI. The training samples used in this experiment can be seen in Figures 6 and 9c. The trained model achieved 72% of mean average precision (mAP). Although the mAP value is lower, it is evident that this model is incapable of learning the negative flow class which is the under sampled class. The individual average precision values are 78%, 60%, and 78% for zero, negative, and positive flows, respectively. These led us to the next experiment with data augmentation by flipping the MFI.

Table . Experiment parameters and (mean) average precision evaluation results in training and testing sets. Zero-flow, negative-flow, and positive-flow are abbreviated as ZF, NF, and PF, respectively. Yellow highlighted experiments are conducted with the proposed method. YOLOv3 weights trained on ImageNet dataset is used as initial weights in pretrained experiment.

Experiment	mAP (testing)	AP (testing)	mAP (training)	AP (training)	Anchors (hxw)	Image size (hxw)
Frame	90%	-	99%	-	default	416 × 416
Frame (pretrained)	93%	-	99%	-	default	416 × 416
Black	73%	ZF: 71% NF: 74% PF: 75%	99%	ZF: 99% NF: 99% PF: 99%	24 × 40 24 × 120 24 × 350	64 × 1024
Base	72%	ZF: 78% NF: 60% PF: 78%	99%	ZF: 99% NF: 99% PF: 99%	24 × 40 24 × 120 24 × 350	64 x 1024
Flip	83%	ZF: 85% NF: 82% PF: 81%	98%	ZF: 97% NF: 99% PF: 99%	24 × 40 24 × 120 24 × 350	64 × 1024
Deep-sort	66%	ZF: 66% NF: 65% PF: 68%	79%	ZF: 78% NF: 76% PF: 81%	-	416 × 416
Deep-sort (pretrained)	68%	ZF: 70% NF: 66% PF: 69%	80%	ZF: 79% NF: 78% PF: 82%	-	416 × 416

Training on MFI with augmentation by flipping (flip): The final experiment with MFI is adding the flipped images onto the base MFI. Training with this model achieved 83% mean average precision, and the average precision is 85%, 82%, and 81% for zero, negative, and positive flows, respectively. Most of the false positives are multiple detections near $abs(\theta) \approx 15$ that is the boundaries of negative and positive flow classes with the zero flow. The overall mean average precision is lower than frame based vehicle detection. It indicates there is an additional temporal variation in motion than vehicle shape, which can be overcome by adding more data to the training stage. Additionally, truck boxes are not as precise as car and bus boxes.

Motion classification based on tracking (deep-sort): Deep-Sort is one of the state-of-the-art tracking methods. We use it as a baseline comparison method. We use the frame based weights trained on the TME dataset as a detection stage and use Deep-Sort's weights for the tracking modules. The detected and tracked bounding boxes are used to compute three class motion classification according to the same equation in (4) with setting $dt = 12$ and the same criteria in (3). The mean average precision in the testing set is around 66% with 66%, 68%, and 65% for zero, positive, and negative flow classes. The training set mAP cannot reach 99% because of imprecise tracking bounding boxes with pretrained weights on the deep-sort model. The overall precisions are lower than our proposed method. Also, any tracking based motion classification method is bounded by the average precision of a vehicle detection method in video frames which is 93% in this case. As a side note, the trained model on the TME with YOLOv3's pretrained slightly increased deep-sort's mAP to 68% and 80% for testing and training sets, respectively.

5. Conclusion

This paper introduced a new unified motion based method to detect and classify vehicles in driving videos. The method is tested on publicly available driving videos in highway scenarios and reaches a mean average precision of 83% that is better than the deep-sort method. Several frames are unified into MFI to minimize the processing data without losing shape features. A traditional shape-based object detector is applied to the MFI for real-time processing. Vehicle motion is classified into three classes based on its motion direction requiring no additional postprocessing, like tracking bounding boxes across frames. In the future, several motion profile patch images can be used in MFI to cover all the vehicles in the environment if missed due to the change in camera viewing direction. Furthermore, quadrangle type bounding boxes or semantic segmentation can be used instead of axis aligned bounding boxes to better fit motion trajectories.

Acknowledgment

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Contribution of authors

M.K. designed conceptualization, applied methodology, prepared/wrote original draft, used software and supervised, T.T. constructed data curation, wrote, reviewed and edited, used software and investigated.

References

[1] Kilicarslan M, Zheng JY. Visualizing driving video in temporal profile. In: IEEE Intelligent Vehicles Symposium; Michigan, USA; 2014. pp. 1263-1269.

[2] Kilicarslan M, Zheng JY. Predict vehicle collision by TTC from motion using a single video camera. IEEE Transactions on Intelligent Transportation Systems 2019; 20 (2): 522-533. doi: 10.1109/TITS.2018.2819827

[3] Redmon J, Farhadi A. YOLOv3: an incremental improvement. arXiv 2018; arXiv:1804.02767.

[4] Caraffi C, Vojir T, Trefný J, Šochman J, Matas J. A system for real-time detection and tracking of vehicles from a single car-mounted camera. In: IEEE Intelligent Transportation Systems; Anchorage, AK, USA; 2012. pp. 975-982.

[5] Yuan C, Yang C, Xu Z. Simple vehicle detection with shadow removal at intersection. In: Second International Conference on Multimedia and Information Technology; Kaifeng, China; 2010. pp. 188-191.

[6] Leeuwen V, Groen CAG. Vehicle detection with a mobile camera: Spotting midrange, distant, and passing cars. IEEE Robotics & Automation Magazine 2005; 12 (1): 37-43. doi: 10.1109/MRA.2005.1411417

[7] Cheon M, Lee W, Yoon C, Park M. Vision-based vehicle detection system with consideration of the detecting location. IEEE Transactions on Intelligent Transportation Systems 2012; 13 (3): 1243-1252. doi: 10.1109/TITS.2012.2188630

[8] Betke M, Haritaoglu E, Davis L. Real-time multiple vehicle detection and tracking from a moving vehicle. Machine Vision and Applications 2000; 12: 69-83. doi: 10.1007/s001380050126

[9] Bertozzi M, Broggi A, Castelluccio S. A real-time oriented system for vehicle detection. Journal of Systems Architecture 1997: 43 (1): 317-325. doi: 10.1016/S1383-7621(96)00106-3

[10] Bensrhair A, Bertozzi A, Broggi A, Fascioli A, Mousset S et al. Stereo vision-based feature extraction for vehicle detection. In: IEEE Intelligent Vehicle Symposium; Versailles, France; 2002. pp. 465-470.

[11] Zielke T, Brauckmann M, Vonseelen W. Intensity and edge-based symmetry detection with an application to car-following. CVGIP: Image Understanding 1993; 58 (2): 177-190. doi: 10.1006/ciun.1993.1037

[12] Bücher T, Curio C, Edelbrunner J, Igel C, Kastrup D et al. Image processing and behavior planning for intelligent vehicles. IEEE Transactions on Industrial Electronics 2003; 50 (1): 62-75. doi: 10.1109/TIE.2002.807650

[13] Tsai LW, Hsieh JW, Fan KC. Vehicle detection using normalized color and edge map. IEEE Transactions on Image Processing 2007; 16 (3): 850-864. doi: 10.1109/TIP.2007.891147

[14] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition; San Diego, CA, USA; 2005. pp. 886-893.

[15] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition; Kauai, HI, USA; 2001. pp. 511-518.

[16] Neumann D, Langner T, Ulbrich F, Spitta D, Goehring D. Online vehicle detection using Haar-like, LBP and HOG feature based image classifiers with stereo vision preselection. In: IEEE Intelligent Vehicles Symposium; Los Angeles, CA, USA; 2017. pp. 773-778.

[17] Li X, Li L, Flohr F, Wang J, Xiong H et al. A unified framework for concurrent pedestrian and cyclist detection. IEEE Transactions on Intelligent Transportation Systems 2017; 18 (2): 269-281. doi: 10.1109/TITS.2016.2567418

[18] Tian Y, Luo P, Wang X, Tang X. Deep learning strong parts for pedestrian detection. In: IEEE International Conference on Computer Vision; Santiago, Chile; 2015. pp. 1904-1912.

[19] Girshick R. Fast R-CNN. In: IEEE International Conference on Computer Vision; Santiago, Chile; 2015. pp. 1440-1448.

[20] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S et al. SSD: Single shot multibox detector. In: Spring Lecture Notes in Computer Science 2016 European Conference on Computer Vision; Amsterdam, Netherlands; 2016. pp. 21-37.

[21] Horn BKP, Schunck BG. Determining optical flow. Artificial Intelligence 1981; 17 (1): 185-203. doi: 10.1016/0004-3702(81)90024-2

[22] Smith SM, Brady JM. ASSET-2: real-time motion segmentation and shape tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 1995; 17 (8): 814-820. doi: 10.1109/34.400573

[23] Cao Y, Renfrew A, Cook P. Vehicle motion analysis based on a monocular vision system. In: IET Road Transport Information and Control Conference; Manchester, UK; 2008. pp. 1-6.

[24] Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. In: IEEE International Conference on Image Processing; Beijing, China; 2017. pp. 3645-3649.

[25] Jazayeri A, Cai H, Zheng YZ, Tuceryan M. Vehicle detection and tracking in car video based on motion model. IEEE Transactions on Intelligent Transportation Systems 2011; 12 (2): 583-595. doi: 10.1109/TITS.2011.2113340

[26] John V, Mita S. Vehicle semantic understanding for automated driving in multiple lane urban roads using deep vision-based features. In: International Joint Conferences on Artificial Intelligence; Macao, China, 2019. pp. 1-7.

[27] Hou L, Xin L, Li SE, Cheng B, Wang W. Interactive trajectory prediction of surrounding road users for autonomous driving using structural-LSTM network. IEEE Transactions on Intelligent Transportation Systems 2019; 21 (11): 1-11. doi: 10.1109/TITS.2019.2942089

[28] Geiger A, Lenzz P, Urtasun R. Are we ready for autonomous driving? the KITTI vision benchmark suite. In: IEEE Computer Vision and Pattern Recognition; Providence, RI, USA; 2012. pp. 3354-3361.

[29] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M et al. The cityscapes dataset for semantic urban scene understanding. In: IEEE Computer Vision and Pattern Recognition; Las Vegas, LA, USA; 2016. pp. 3213-3223.