2015 IEEE 18th International Conference on Intelligent Transportation Systems

A Real-time Multi-scale Vehicle Detection and Tracking Approach for Smartphones

Eduardo Romera, Luis M. Bergasa, Roberto Arroyo

***Abstract*—Automated vehicle detection is a research field in constant evolution due to the new technological advances and security requirements demanded by the current intelligent transportation systems. For these reasons, in this paper we present a vision-based vehicle detection and tracking pipeline, which is able to run on an iPhone in real time. An approach based on smartphone cameras supposes a versatile solution and an alternative to other expensive and complex sensors on the vehicle, such as LiDAR or other range-based methods. A multi-scale proposal and simple geometry consideration of the roads based on the vanishing point are combined to overcome the computational constraints. Our algorithm is tested on a publicly available road dataset, thus demonstrating its real applicability to ADAS or autonomous driving.**

I. Introduction

In 2014, more than $25,700$ people died on the roads of the European Union [1]. Studies about accident causation, like NHTSA's [2], attribute 94% of accidents to driver-related reasons, such as distraction or inattention. Therefore, research and development of prevention measurements focused on drivers is essential for reducing fatality on the roads.

In this direction, researchers and manufacturers have significantly progressed in the development of algorithms and systems that are able to perceive the vehicle environment with inference capabilities that are close to human ones. On the one hand, systems that learn about the static environment (i.e. road markings, traffic signs) have been studied in the state of the art with remarkable results [3]. On the other hand, the dynamic environment (i.e. pedestrians, vehicles) is still a challenging aspect, due to the high variability of the objects to be avoided.

Over the last decades, sensors like RADAR and LiDAR have been widely studied as a solution to this problem. However, the high cost and space constraints of these sensors have situated computer vision as one of the most common alternatives [4]. Cameras have become cheaper, smaller and of higher quality than ever before. In addition, computational costs associated with computer vision have been reduced due to the improvements in processing units.

Specifically, today smartphones have the computing capabilities of a full computer from few years ago and a high market penetration. These devices provide good embedded units to solve computer vision problems because of their integrated cameras and their powerful processing and communication capabilities. That is why in the last years there has been an active work on using smartphones as a low-cost platform for monitoring and assisting drivers.

Fig. 1. DriveSafe App running in a real environment.

In previous work [5], authors introduced DriveSafe, a smartphone app that detects and alerts inattentive driving behaviours by making use of the camera, microphone, GPS, and inertial sensors. This supposes an affordable way to provide safety features that are normally only available in top-end vehicles. With the aim of expanding the analysis about dangerous behaviours (e.g. tailgating), we present an algorithm for ahead vehicle detection and tracking that is integrated in DriveSafe application (see Fig. 1). Our proposal is based on a multi-scale approach that takes into consideration the road geometry to overcome the computational constraints. The algorithm is evaluated on a publicly available motorway dataset [6], demonstrating its viability for Advanced Driver Assistance Systems (ADAS) and autonomous driving applications.

II. Related work

Vision-based object detection has been widely studied over the last years. One of the key works that supposed a breakthrough is the Viola-Jones algorithm [7], which is based on a sequential classifier with Haar-like features that demonstrated real-time performance on the face detection problem. Since then, researchers have proposed several approaches based on multiple classification algorithms (SVM [8], AdaBoost and variants [9], [6]) and varied features (Haar [10], LBP [6], HOG [8], ICF [9] and ACF [11]).

*This work was supported in part by the MINECO Smart Driving Applications (TEC2012-37104) project and by the RoboCity2030 III-CM project (S2013/MIT-2748) funded by Programas de I+D en la Comunidad de Madrid and cofunded by Structured Funds of the UE.

The authors are with the Department of Electronics, Universidad de Alcalá (UAH). Alcalá de Henares, Madrid, Spain. e-mail: eduardo.romera@edu.uah.es, roberto.arroyo@depeca.uah.es, luism.bergasa@uah.es

978-1-4673-6596-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ITSC.2015.213

On the specific research area of vehicles, detection in static images is a widely studied topic. The recent appearance of vehicle-annotated motorway datasets (i.e. LISA [12] and TME Motorway [6]) has allowed to evaluate detection performance on real driving environments. Thus, performance is measured not only as a theoretical amount of true positives in a set of images but as real execution in a dynamic motorway environment, where tracking and filtering techniques are as important as detection.

In this way, there have been several recent works that have focused on developing improved tracking algorithms by making minimal changes to the Viola-Jones detection framework. Particle Filtering (PF) has been a widely used technique. An example based on PF is [13], which proposes a full integration between lane and vehicle tracking modules to complement each other. In [6], an algorithm based on Flock of Trackers and a Learn and re-detect module is implemented following TLD method (Tracking-Lerning-Detection). The work in [14] proposes the use of a Probability Hypothesis Density (PHD) filter to track features detected within the bounding box of the vehicle, with some pruning techniques to counteract the high computational requirements of the filter.

Additionally, the motorway environment allows to apply expert knowledge from the road. Several works focus on applying geometrical constraints, mostly to satisfy real-time requirements by pruning the detection search window. For instance, methods like [8] implement an adaptative coarse-to-fine object search to restrict possible scan-ROI positions. This work also applies a multi-scale feature preprocessing stage to award more resolution to distant ROIs and reduce processed pixels over 50%, similarly to what is done in the present work. The approach in [9] proposes reducing scales per octave depending on uncertainty from tracked object and giving computation priority to near vehicles. Both works use variants of AdaBoost that early reject regions with low object probability: Boosted Cascade in [8] and Soft-Cascade in [9].

With the aim of satisfying both computational and detection requirements, we combine multi-scale approaches with low-computational tracking methods. Simple considerations of the road geometry are taken into account by means of a lane detector. The result is an efficient algorithm that is suitable for smartphones. At the end of the document, it is evaluated on a motorway dataset to prove its performance on a real environment.

III. System design

This work is focused on developing a vehicle detection and tracking algorithm suitable for smartphones. Hence, a key aspect in the system is the computational efficiency. As seen in the state of the art, scanning a full image with a classifier is an expensive operation. Thus, applying knowledge from the road geometry to prune the search window is more efficient than scanning the image by brute force. Considering any road environment, vehicles in the scene may either appear from behind as near vehicles (if they move faster than us) or either be approached from far (if they move slower than us or they

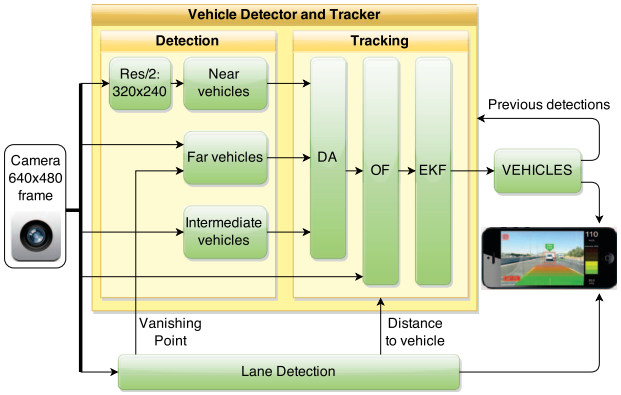


Fig. 2. Full system diagram. DA, OF, and EKF stand for Data Association, Optical Flow, and Extended Kalman Filter respectively.

are in the opposite direction lane). With this assumption, we propose a multi-scale approach that optimizes the discovery of new vehicles by making use of different detection windows.

The detection can be divided into three main stages, as can be seen in Fig. 2. They are based on an AdaBoost classifier. Firstly, a large image patch with low resolution is scanned by the detector to mainly discover near vehicles (Sec. III-A). Secondly, a small image patch in the surroundings of the vanishing point is scanned at a higher resolution in order to discover far vehicles (Sec. III-B). Thirdly, the detector scans specific image patches corresponding to previously discovered vehicles in order to reaffirm detections and to cover those that do not fit in the near nor the far scan windows (Sec. III-C). Fig. 3 shows a video-frame where each of the stages is relevant to perform the detection of a specific vehicle in the scene.

Data association between detections in each frame and their corresponding tracked candidates is decided by a simple overlapping formula (Sec. III-D). Position estimation is enhanced with Optical Flow (Sec. III-E). Tracking is performed by an Extended Kalman Filter (Sec. III-F). Lane detection (Sec. III-G), which is performed by DriveSafe App, is seized to enhance the detection by providing estimated distance to the car and road vanishing point.

The whole implementation is oriented towards simplicity in order to fulfil real-time constraints. Thus, the input image is reduced to 640x480, from which all detection windows are extracted. All stages are performed in every frame.

A. Detector 1: Near window

In this stage, the image resolution is resized to a half resolution (320x240). This extremely reduces computational cost in the AdaBoost detector, although it has two negative consequences. Firstly, the detector recall rate is reduced for vehicles that are farther than a distance (≈30m), as the lower resolution diffuses the vehicle features. Secondly, the training process is done with a fixed model size (20 pixels in our case), so the objects with a pixel width lower than this threshold will not be detected by the classifier. Therefore, the function of this detection ROI is only to detect near vehicles, leaving those that are further than 30 meters to the intermediate or far window.

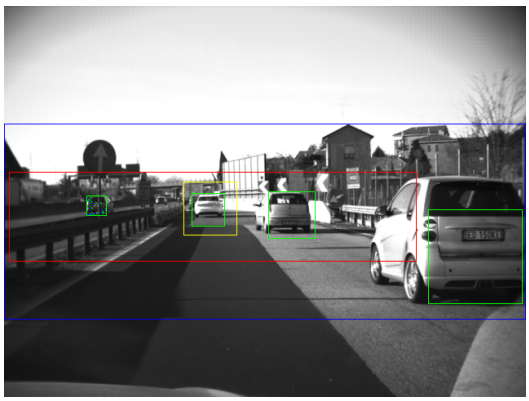


Fig. 3. [Best viewed in color] Example of the three vehicle detection stages working. Green boxes represent detections, blue the near-window, red the far-window, yellow the intermediate window (vehicle-specific) and the points depicted in blue in the vehicle of the left represent Optical Flow.

Additionally, the sky is filtered out to avoid unnecessary computation. Considering that the camera will be perpendicular to the ground, the horizon provided by DriveSafe is approximately situated at half of the image. With the aim of leaving a margin for near cars or trucks, only about 30% of the top is removed. About 20% of the bottom of the image is also removed in order to avoid the car bonnet (estimated by DriveSafe) and a small part of the nearby road that does not affect detection. Thus, a total of around 50% of the vertical size of the image is removed, reducing considerably the computation cost of this stage.

B. Detector 2: Far window (zoom)

The far window covers the deficits of the previous stage for detecting distant vehicles. Thus, a small patch is extracted from the input image in the neighbourhood of the vanishing point, which is provided in our system by the lane detector module of DriveSafe (Sec. III-G). This approach using the vanishing point of the image supposes an improvement over using a fixed window in the half of the image in cases of curved roads.

The size of this window must be small enough to have a light computational cost and large enough to fit most far vehicles Therefore, this size is experimentally set to 500x110 pixels (see red rectangle in Fig. 3).

Considering that this far window is overlapped within the near one, this region will be scanned twice by the detection. Thus, some vehicles could be detected by both detectors. In these cases, the score of the AdaBoost detector (a weighted sum of the weak classifiers) is used to determine which of both detections is predominant.

C. Detector 3: Intermediate window

Once a vehicle has been discovered by any of the previous windows, a third detection is performed on the surroundings of the last position of the car. This ensures re-detection independently of road position, covering intermediate areas where the other windows have deficiencies. Thus, it reaches all those vehicles that have moved far enough to not be correctly detected by the near window but are not far enough to fit inside the far window.

The intermediate window is vehicle-specific. It is focused on the surroundings of the previously known vehicle position. An extra size is given by a margin of half the width and height on each of the sides of the previously detected bounding box. The image used to extract this window is the full 640x480 one. As it will be seen in Sec. IV-C, the computing cost of this window is extremely low.

D. Data association

In order to associate new detections in the current frame and tracked candidates, an overlap matrix is computed as follows:

$$overlap = \frac{area(BB_i \cap BB_j)}{area(BB_i \cup BB_j)} \quad (1)$$

where BB is the bounding box of new vehicle i and tracked candidate j. Vehicles are associated if the overlap is higher than a certain threshold.

When a new vehicle is discovered and it does not overlap with any previous candidate, it is saved as a new candidate if it is detected by either the near-window or the far-window at least twice in the last three frames. This significantly reduces false positives and avoids that these detections are kept in the system by the tracking algorithm. When a candidate does not have a new detection associated in the current frame, this is moved using optical flow of local features.

E. Optical Flow

Local features from detected vehicles are used to enhance tracking. On detection, a small image patch containing the vehicle is saved. This image patch is used to compute features that are searched in the next frame in the proximity of the last known position. Thus, corners are computed by Shi-Tomasi method and are tracked using the optical flow obtained by Lucas-Kanade algorithm [15]. The median of the flow from all found points is used to compute the vehicle motion, which is used to correct the previously known position of an unpaired candidate in the current frame. When less than ten points have been tracked by the algorithm, the candidate is considered not valid and removed. This step has low computational cost, as the corner search happens only in a reduced region.

F. Extended Kalman Filter

Tracking is performed with an Extended Kalman Filter (EKF). The pinhole camera model is used to set the following 3D states of each candidate: lateral position, longitudinal position (distance to camera), vehicle width (detection pattern is square), and their respective derivatives. 3D measurements are calculated by the lane detection module of DriveSafe.

This filter is essential to keep a stable detection and avoid the flicker produced when the detection is noisy or when there is a change between windows in the multi-scale approach.

G. Lane detection

Lane detection is carried out by DriveSafe App to enhance vehicle detection by providing road information. Reader may refer to [5] in order to learn more about implementation.

IV. RESULTS

A Gentle AdaBoost classifier is trained based on LBP. Haar was also tested and demonstrated much slower performance with no significant improvement. HOG, ICF and ACF were discarded due to computational constraints. The classification is performed on 20x20 pixel patterns. This defines the model size, and thus the minimum vehicle width that can be detected in the image. This value affects at the image resolution used by the detector (640x480). In the charts, pixel values are translated to 1024x768 for comparison with the original dataset, so the minimum vehicle width is shown as 32 pixels.

All images used for the training are publicly available and they correspond to a mixture between public datasets. Positives are only obtained from GTI dataset [16], corresponding to 3425 images of rears of cars and trucks. Negatives are obtained from multiple datasets: GTI [16], KITTI [17], Caltech Cars (Rear) background [18] and GRAZ-02 [19] (only images of background without cars), forming a total of 13868 negatives.

To the best of our knowledge, there are only two publicly available vehicle-annotated datasets for motorway video sequences: LISA Vehicle Detection [12] and TME Motorway [6]. The first includes 2 motorway sequences for a total duration of one minute, while the latter is formed by 28 motorway clips for a total of approximately 27 minutes (30000+ frames). We present the results on the larger one (TME), as it allows evaluating the performance in a real motorway situation. Recent works on vehicle detection have presented results on it: [6], [8], [9] and [14] (this one only presents results about tracking error). It contains two subsets: "Daylight", which is larger and its light conditions might be more common in daily situations, and "Sunset", which was recorded in challenging conditions where the sun is low and its light significantly affects visibility, allowing the evaluation of the algorithm robustness.

A. Detection performance

In Figs. 4 and 5 we present the evaluation statistics collected for the algorithm performed on the dataset. As explained in [6], the dataset's approximate ground truth (GT') was obtained from laser scans, hence the reliability limitation beyond 60-70 meters, when less than 3 laser reflections per vehicle become available. The results are presented in the same format and evaluation process as the original paper, for comparison.

Results show that precision remains over 90% for both dataset subsets (Daylight and Sunset). Recall rate is over 95% for near cars and it remains very high until 60m. The difference in performance between trucks and cars corresponds to the lower proportion of truck samples that is used in the training set. Ideally, cars and trucks should be detected by different classifiers due to the remarkable model difference, but this would not be a computationally efficient solution. Thus, both are used together to train the same classifier, which results in an acceptable overall performance.

Despite the fact that the GT' is less reliable over 60m, the algorithm detection rate decays over this distance due to the low resolution that is kept for computational constraints. Our work is focused on providing high detection rates within a reasonable distance. Ahead vehicles under 60m have the most impact on driving behaviour, and are the ones to be utterly considered by ADAS or autonomous driving algorithms.

As can be seen in Fig. 5, detection performance remains similar in the more challenging "Sunset" subset. This demonstrates that the algorithm is robust against light variations. Additionally, further tests in a real driving environment have also demonstrated robustness under severe rain conditions, although we cannot provide quantitative results due to the lack of ground truth.

B. Comparison with related works

The objective of this work was not to improve the detection performance of an existing system, but obtain similar results with state-of-the-art works, considering the smartphone computational constraints. In Fig. 6, we show our results compared to state-of-the-art methods that made use of this dataset: [8] (Boosted Cascade + Haar + LRF + Road constraints), [6] (WaldBoost + LBP + FoT) and [9] (SoftBoost + ICF).

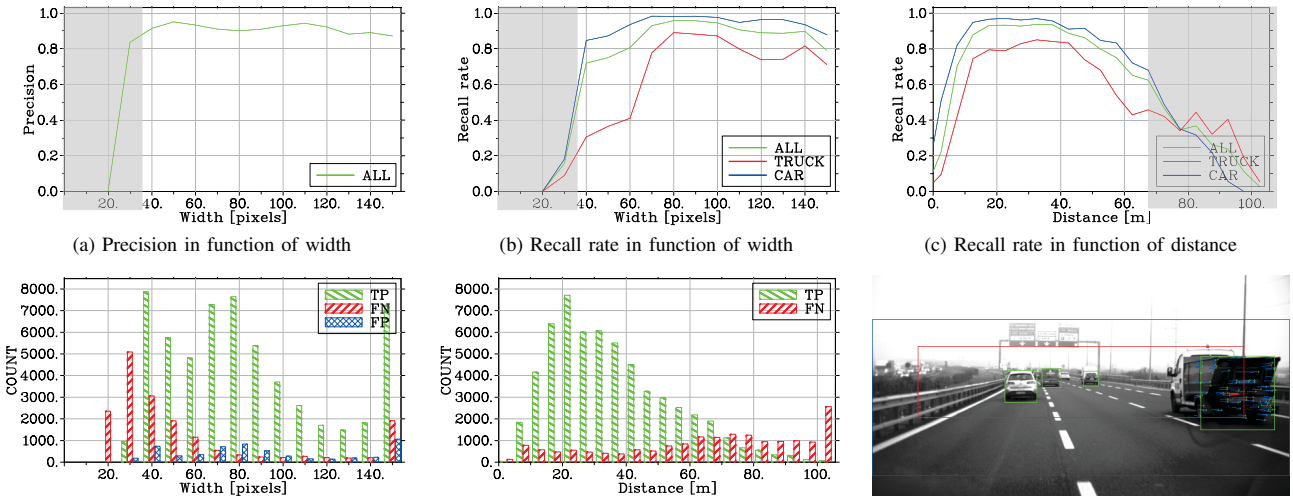
As can be seen in Fig. 6a and 6c, the precision of our algorithm is slightly lower than other results because high recall has been preferred in the trade-off between precision and recall, as we give more importance to detect all cars in a range and noisy detections are easier to filter with further lane analysis. Fig. 6b and 6d show that recall rate is higher than other state-of-the-art works until 60m. Additionally, the results for "Sunset" show that our algorithm is the least affected by bad light conditions.

C. Computing performance

Table I shows the computation time for each of the modules of the detection algorithm for three different devices (iPhone5, iPhone6, and iPhone Simulator). The most expensive stage is the far-window due to its high resolution, even though it covers a smaller region. This demonstrates the advantage of multi-scaling, as a single full-resolution window would be inviable.

The detector runs inside our DriveSafe application, simultaneously with the rest of its ADAS modules. The hermetic character of iOS does not allow to isolate the application from other unrelated background processes, so it was not possible to guarantee full CPU dedication to the algorithm. This produces a slight decay in the expected performance. Nevertheless, it supposes a more realistic evaluation, since the results obtained correspond to an application running in the smartphone OS along with other several routines (e.g. call management), which is the case for the average user.

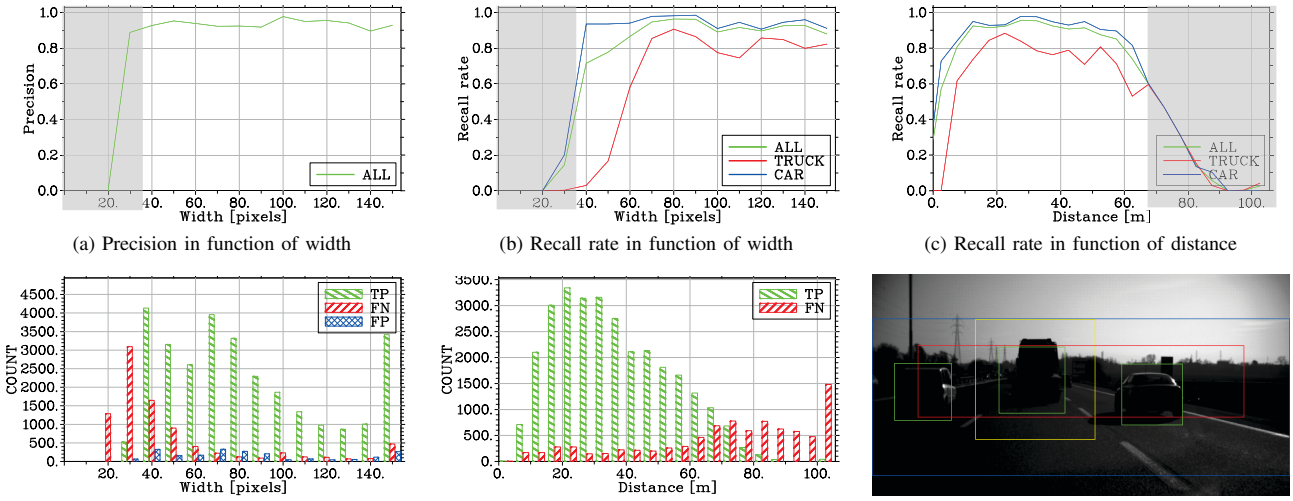
These results represent the total time needed by one CPU core to process one frame. Considering the camera has a rate of 30 fps, not all frames are processed. In DriveSafe App, the detection algorithm runs as a medium-priority thread providing detections. From experimental tests in a motorway environment, we have inferred that 5-10 fps is enough for a robust detection performance. Running the algorithm on the TME dataset with lower frame rates (5 and 10 Hz) by skipping frames has also produced similar results to those presented in



(a) Precision in function of width (b) Recall rate in function of width (c) Recall rate in function of distance

(d) All detections, grouped by width (e) All detections, grouped by distance (f) Representative image from subset

Fig. 4. [Best viewed in color] Detection algorithm evaluated on the "Daylight" subset of the TME Dataset. A grey box is placed over the chart region where the GT' is not considered reliable due to a laser limitation. Vehicle pixel width corresponds to a resolution of 1024x768, for comparison with original charts.



(a) Precision in function of width (b) Recall rate in function of width (c) Recall rate in function of distance

(d) All detections, grouped by width (e) All detections, grouped by distance (f) Representative image from subset

Fig. 5. [Best viewed in color] Detection algorithm evaluated on the "Sunset" subset of the TME Dataset.

Fig. 4 and 5. The worst case scenario are vehicles that overtake or brake suddenly near the camera sides, thus producing the largest motion in image pixels. Even in these cases, the relative speed between the detected vehicle and the camera carrier has to be immensely high to produce a loss in the tracking process.

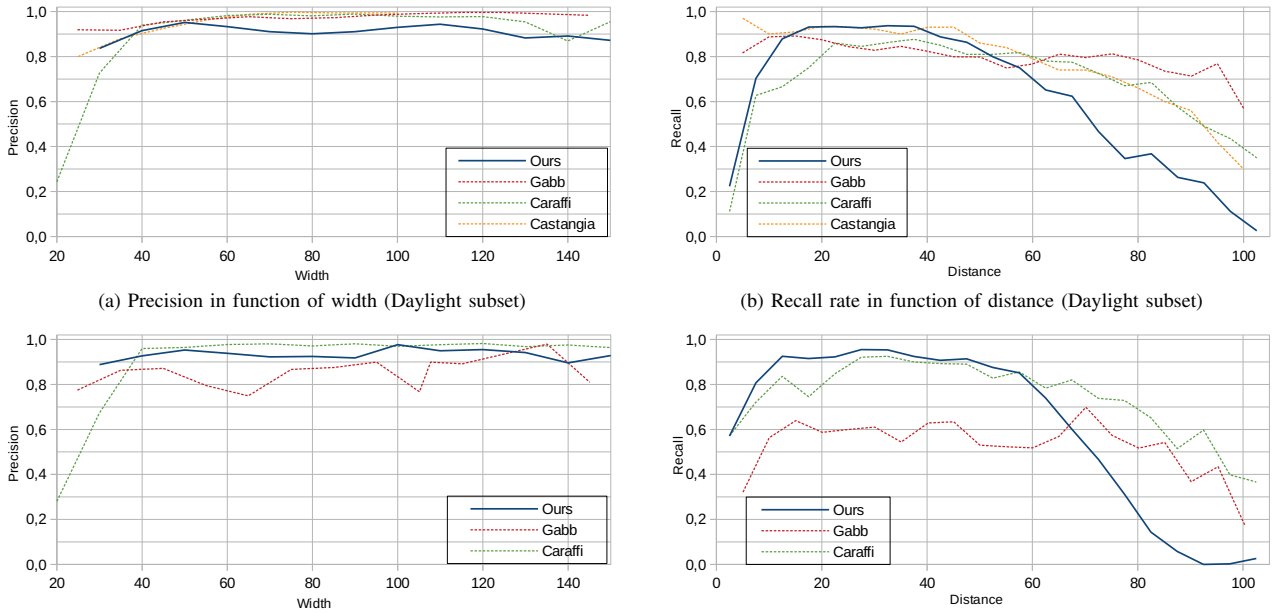
Results are also provided for an iPhone 6 simulator run by an Intel Core i7@2.2GHz processor for comparison with other works. Although the simulator supposes a significant computational overhead and the algorithm would achieve higher frame rates as a standalone code, it doubles the standardized real-time rate of 30 fps. This demonstrates its portability to other devices such as in-vehicle computers.

AVERAGE COMPUTATION TIME [ms]			
Module	iPhone 5	iPhone 6	Simulator (i7)
Near-window	48	26	6
Far-window	70	41	8
Intermediate-win.	6	4	1
Optical Flow	3	2	<0.5
Extra operations	5	3	<0.5
TOTAL	132	76	16
FPS	7.6	13.2	62.5

TABLE I
AVERAGE COMPUTATION TIME FOR DETECTION MODULES.

V. Conclusion

We have presented a vehicle detector and tracker that can run in real time on an iPhone, in parallel with other ADAS processes such as a lane detector. A multi-scale approach and geometrical considerations have been used to overcome the computational constraint. The result is a versatile algorithm that can be implemented on any smartphone and performs well on motorway environments without the requirement of large training sets. Experiments on a publicly available motorway dataset demonstrate that our detection performance is similar



(a) Precision in function of width (Daylight subset)

(b) Recall rate in function of distance (Daylight subset)

(c) Precision in function of width (Sunset subset)

(d) Recall rate in function of distance (Sunset subset)

Fig. 6. [Best viewed in color] Comparative charts of state of the art results for "ALL" (cars and trucks) between our approach ("Ours"), results presented in [8] ("Gabb"), in [6] ("Caraffi") and in [9] ("Castangia"). Results of Sunset subset are not available in [9].

to state-of-the-art works and it is robust against light changes. Its low computational cost allows an efficient integration in ADAS or autonomous vehicles.

Future work will involve producing a higher-level of driving behaviour analysis based on the vehicle detection and the techniques applied in [5]. Improvements of the detection and tracking algorithm could be based on a higher integration of the lane and vehicle modules to complement each other in a more complex way. In addition, other preprocessing stages could be added for minimizing issues associated with shadowing or sunset conditions in vehicle detection, such as the illumination invariance techniques presented in [20].

REFERENCES

[1] *Road safety in the European Union: Trends, statistics and main challenges*. European Comission, March 2015. [Online]. Available: http://ec.europa.eu/transport/road_safety/pdf/vademecum_2015.pdf

[2] N. H. T. S. Administration *et al.*, "National motor vehicle crash causation survey (dot hs 811 059)," *Washington, DC*, 2008.

[3] J. C. McCall and M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE Trans. on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 20–37, 2006.

[4] S. Sivaraman and M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[5] L. M. Bergasa, D. Almería, J. Almazán, J. J. Yebes, and R. Arroyo, "Drivesafe: an app for alerting inattentive drivers and scoring driving behaviors," in *IEEE Intelligent Vehicles Symp. (IV)*, 2014, pp. 240–245.

[6] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matası, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," in *IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2012, pp. 975–982.

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. 511–518.

[8] M. Gabb, O. Lohlein, R. Wagner, A. Westenberger, M. Fritzsche, and K. Dietmayer, "High-performance on-road vehicle detection in monocular images," in *IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2013, pp. 336–341.

[9] L. Castangia, P. Grisleri, P. Medici, A. Prioletti, and A. Signifredi, "A coarse-to-fine vehicle detector running in real-time," in *IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2014, pp. 691–696.

[10] A. Haselhoff and A. Kummert, "A vehicle detection system based on haar and triangle features," in *IEEE Intelligent Vehicles Symp. (IV)*, 2009, pp. 261–266.

[11] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[12] S. Sivaraman and M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Trans. on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 267–276, 2010.

[13] ——, "Integrated lane and vehicle detection, localization, and tracking: A synergistic approach," *IEEE Trans. on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 906–917, 2013.

[14] F. Garcia, A. Prioletti, P. Cerri, A. Broggi, A. de la Escalera, and J. Armingol, "Visual feature tracking based on phd filter for vehicle detection," in *Int. Conf. on Inf. Fusion (FUSION)*, 2014, pp. 1–6.

[15] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, pp. 1–10, 2001.

[16] J. Arróspide, L. Salgado, and M. Nieto, "Video analysis-based vehicle detection and tracking using an mcmc sampling framework," *EURASIP Journal on Advances in Signal Processing*, no. 1, pp. 1–20, 2012.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[18] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2003, pp. 264–271.

[19] M. Marszatek and C. Schmid, "Accurate object localization with shape masks," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[20] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Towards life-long visual localization using an efficient matching of binary sequences from images," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.