Computer vision algorithms for detecting secondary tasks in naturalistic driving studies

F. Dellinger [1*], M. Robert-Seidowsky [1], E. Bernard [1], L. Guyonvarch [2], A.Guillaume [2]

[1] WASSA, 5 rue de l'église, Boulogne, France
[2] LAB Renault PSA, 132 rue des Suisses, Nanterre, France
*flora.dellinger@wassa.fr

Keywords: computer vision; driver behavior; distraction detection; phone localization; machine learning; deep learning

Abstract: Driver inattention and distraction are main concerns in road safety studies. In naturalistic driving studies, driver's videos are a valuable source of information and can only be used if annotated. The work presented here is a completely new framework for automatic video annotation. It is based on computer vision methods and focuses on driver distraction related to visual-manual secondary tasks: texting and phone-to-the-ear uses are automatically detected as well as hands on steering wheel and foot behaviour over pedals.

The database used to develop and validate image processing algorithms was recorded by LAB over a six months period. It includes several views showing: the driver face, the driver feet, the entire cockpit and the steering wheel.

The developed framework includes deep learning approaches and HoG+SVM algorithms. It provides driver behaviour indicators over time such as phone-to-the-ear conversation and texting on a phone.

The algorithms performances are presented in this paper. Results are promising in different driving conditions. The next step will be to improve the processing time. The implementation of such processes will be a great opportunity to enhance data processing in future naturalistic studies.

1. Introduction

A crucial question about driver behaviour for road safety is the real evaluation of distraction while driving. The use of nomadic devices (such as mobile phones), becoming increasingly widespread, can be considered as a very important part of secondary tasks [20]. The naturalistic driving studies are the best way to study their impact on driving. Such studies represent a high amount of data because the driver activities are recorded in continuous videos over a long period of time for numerous drivers in naturalistic conditions. Manual annotation of secondary tasks is the easiest way to allow the use of such data, but it is expensive and time-consuming. Thus an automatic detection of defined tasks is an interesting alternative. The challenge is not only on the image processing performances but also on processing time optimization. In this study, we aim to automatically extract specific events linked to secondary tasks. We distinguish verbal-auditory and visual-manual tasks:

- Hands free use of phone (conversation) either using Bluetooth function or hands free headset are examples of verbal-auditory tasks.
- Visual-manual tasks should have a higher impact on driving parameters. Texting on a phone is a very interesting visual-manual task that can be addressed using naturalistic driving data. Phone-to-the-ear conversation also requests a manual activity in addition to the verbal-auditory task.

Automatic video annotation is only a first step to study driver behaviour and should be completed by manual annotations to find correct start and end of the secondary tasks. Three other indicators are of interest for road safety studies:

- Feet position on pedals. We have observed that drivers sometimes put their feet under the pedals for example when driving with cruise control. In case of an emergency braking, the time to reach the right pedal can be increased and the driver may even reach another pedal without noticing it. There is no precise knowledge of the real prevalence of this phenomenon and naturalistic data could provide valuable information.
- Presence of hands on the steering wheel, linked with driver safety margins.
- Presence of a passenger on the front seat. In many cars type, the only automatic detection is whether the seatbelt is locked but no presence detection is performed (unlike driver presence).

In this paper we present a solution for automatic video annotation of secondary task in NDS.

2. State of the art

There are only a few studies conducted on driver distraction analysis. It is unfortunately largely due to the lack of public available data due to privacy concerns. Among the recent available datasets, we can cite the famous Kaggle Competition [1], a computer vision challenge about driver distraction. Unfortunately, we could not participate to this competition or even use the available dataset because the cameras point of view were strongly different from the ones of our study.

About hands detection, Das [11] uses the Aggregate Channel Features (ACF) object detector from the Piotr's Computer Vision Matlab Toolbox (PMT) [2] algorithm. ACF uses LUV color channels and gradient information (magnitude and orientation). A multi-scale sliding-window approach is then employed with an Adaboost classifier for the decision step on the ACF features. This method is tested on the color VIVA dataset [5] and presents interesting results. However, we have made some experiments for hand detection on our dataset thanks to the available Matlab toolbox but we did not get convincing results, certainly because our data are gray-level images. We can also cite several articles from Ohn-Bar on driver distraction [7]. In [13], he proposes a method to analyze hand actions. On a dataset with colored data, HoG (Histograms of Oriented Gradients) [10] features are extracted on three fixed regions around the dashboard representing the potential hands driver locations, and then classified with Support Vector Machines (SVM) [9]. Face detection and tracking with landmarks are also processed to get the level eye opening and obtain gaze information and head pose estimation. Combining information about the hands and gaze allows to classify the driver situation as “two hands on the wheel”, “hand on the instrument panel” or “hand on the gear shift”. In [14], Ranghes presents a multi-cue tracking framework which, first detects hand with ACF detector then tracks it with a median flow tracker and finally uses a pre-trained SVM to classify hands as right or left.

About phone detection algorithms, most of recent works come with the SHRP-2 [4] project: Seshadri [15] proposes a method to detect if the driver is holding a phone near his/her ear. First, the face is detected with the well-known Viola-Jones face detection algorithm [8], next 49 facial landmarks using SDM (Supervised Descend Method) [18] algorithm are computed to determine two regions localized near both ears. Then, HoG features are extracted on these regions and classified with help of two SVM (two classifiers for right and left positions around the head). We have tested this approach on our dataset. Results were promising in optimal conditions but it was too difficult to be generalized for the diversity of situations we are facing (variety of phones, luminosity changes, face orientation…). Deep-learning is employed in [12] to detect the steering-wheel, the face and hands on SHRP-2 and VIVA dataset. Deep ConvNets is used to extract features, then a SVM is applied on the last layer of the neural network for classification. A new network called “MS-FRCNN” is specially designed to detect objects at various scale since hands are very small objects.

To the best of our knowledge, only one system exists about foot behavior analysis. Tran presents in [16] a framework which starts from a gray-level video facing the driver’s feet and tracks them with an optical flow. The temporal foot behavior is learnt by a Hidden Markov Model with help of previous trajectory information and CAN data.

Unlike our gray-level dataset, most of the used videos are colored which is a huge benefit for computer vision tasks. Also, we did not find any article about texting situation detection. Moreover, most of existing methods are “frame by frame detection” while we would like to interpret results as sequences to obtain temporally global information.

3. Available data

The study was carried out using a database recorded by the LAB. During six months, LAB staff drove three different cars in the Paris area. Seven cameras where installed in the cars and CAN data were recorded synchronously. Drivers were asked to drive as naturally as possible even if they knew data were recorded.

3.1. Video data

For secondary tasks automatic detection, four cameras views of inside the car were used (Figure 1). The cockpit was unfocused on purpose to prevent from identifying passengers who were not aware of being recorded. The other views included images of the driver face, a top view showing the steering wheel, the dashboard and the driver’s hands and a view with the three pedals and driver feet.

Relevant information extraction was a great challenge because of images complexity. First, videos were recorded in gray level and infrared cameras for driver face and feet to record even at night. Secondly, the images were highly compressed, had low resolutions and low frame rates (see Table 1). Data were recorded in various conditions (day/night, with several drivers, rainy/sunny days) which was a real challenge for image processing. Images could indeed be saturated in high lighting conditions (sun). Finally, three different cars were used, leading to small variations in cameras positions.

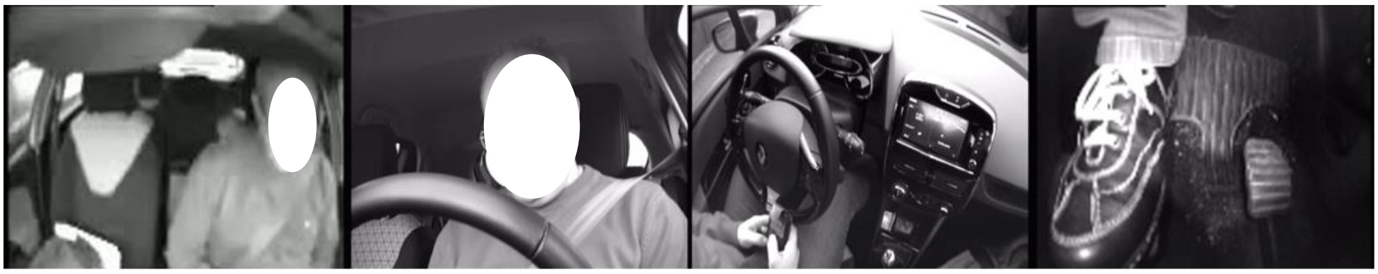


Fig. 1: *Example of the four different views. From left to right, cockpit, driver, top view and pedals cameras.*

3.2. CAN data

In the Renault cars used, CAN network has also been recorded. The car calculators use this network to communicate one with each other and share all the embedded sensor's information. It includes for example measurement of speed, position of pedals and locking of seat belt.

Table 1: Specificities of the different cameras.

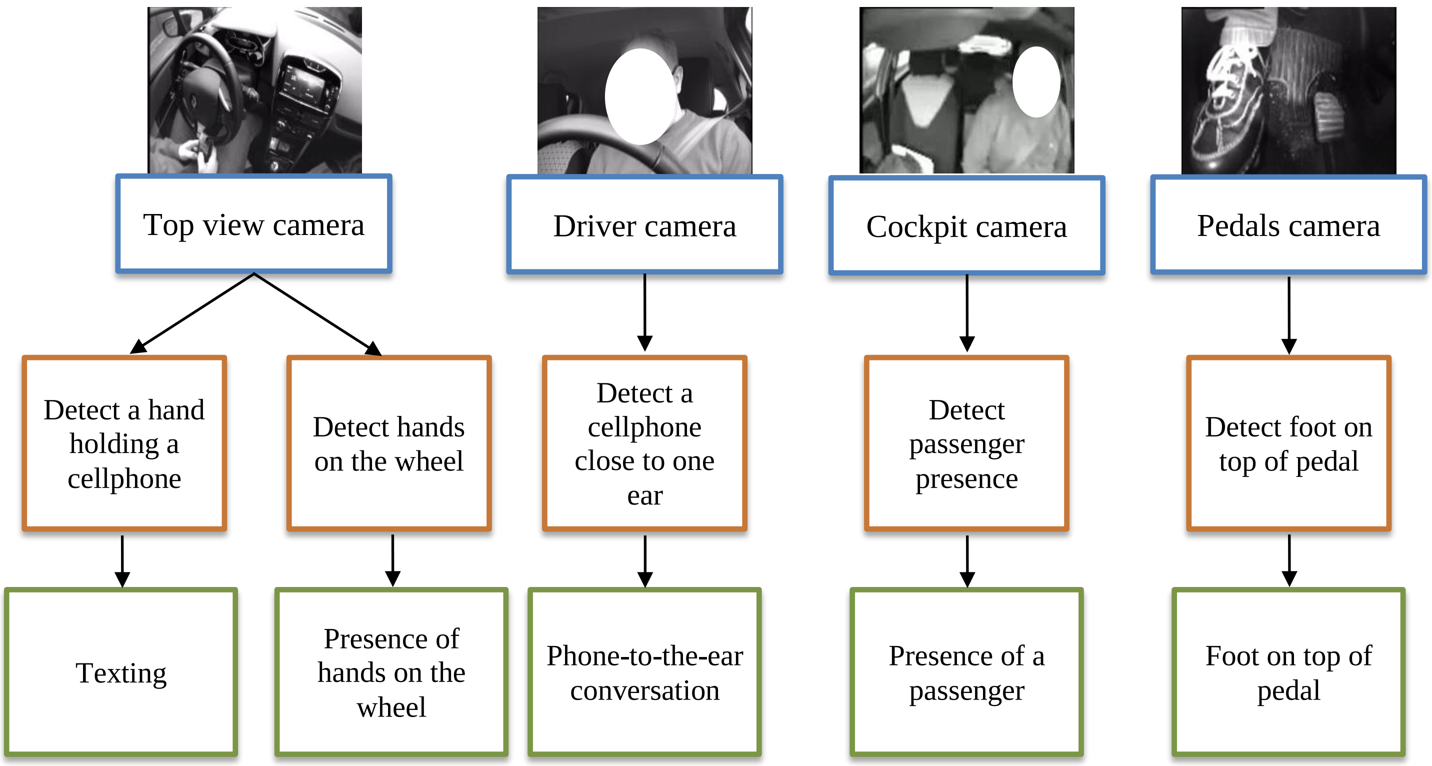
	Cockpit camera	**Driver camera**	**Top view camera**	**Pedals camera**
Frames per second	1	12.5	12.5	25
Resolution (pixels)	160x112	384x288	384x288	160x112

1. Proposed detection algorithms

The automatic detection of secondary tasks is organised as shown in Figure 2. Two types of algorithms are used:

- A classifier that we called **HoG + SVM** for the "Presence of a passenger" and "Foot on top of pedal" detection.
- An algorithm based on **deep learning** for the "Texting", "Presence of hands on the wheel" and "Phone-to-the-ear conversation" detection.

These algorithms are explained in detail in the next subsections.



Top view camera
Driver camera
Cockpit camera
Pedals camera
Detect a hand holding a cellphone
Detect hands on the wheel
Detect a cellphone close to one ear
Detect passenger presence
Detect foot on top of pedal
Texting
Presence of hands on the wheel
Phone-to-the-ear conversation
Presence of a passenger
Foot on top of pedal

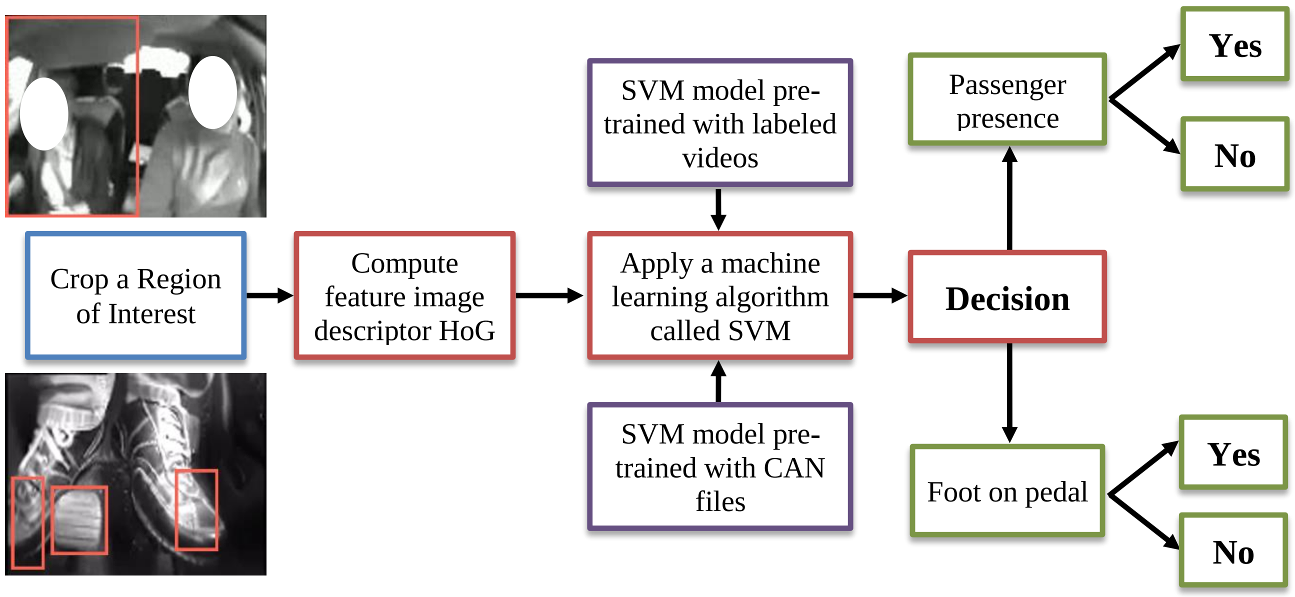
Fig. 2: *Solution framework. For each video (in blue), actions performed on these views are defined (in orange) in order to detect the studied items (in green).*

1.1. HoG + SVM Classifier

This algorithm is common in computer vision for object detection. It is used here for detecting the presence of a front passenger ("Presence of a passenger" detection) and a foot over one of the three pedals ("Foot on top of pedal" detection). Schema of the algorithm is presented in Figure 3.

On each frame, the following steps are executed:

1. A ROI (Region of Interest) is extracted on the image:
   - For the passenger detection, the left half of the cockpit camera view is extracted.
   - For the detection of the foot on pedal, three ROI from the feet camera view are extracted, one at each pedal position. Since this camera view includes strong occlusions and shadows, these ROI were defined manually for each of the three studied cars.
2. A feature image descriptor called HoG (Histogram of Oriented Gradient) is computed on these ROI. This descriptor represents the local shape of objects inside the ROI.
3. Binary classifiers called SVM (Support Vector Machine) are applied to the extracted HoG to obtain binary decisions regarding the presence of objects. These classifiers need to be previously trained with positive and negative samples (images with and without the object).
   - A first SVM model is used to detect a passenger. It is pre-trained with labelled videos.
   - Three other models are used for the detection of the foot on the pedals: one for each pedal. The SVM models are pre-trained for each video using CAN data. Indeed, these files indicate at each frame if one of the pedals is activated, giving access to positive samples. Negative samples are obtained for the brake and gas pedal: we assume there is little chance to find a foot on the gas pedal when the brake pedal is activated, and vice versa. For the clutch pedal, ROI without any foot are extracted manually, since there is no possibility to obtain negative samples through the CAN data. It has to be noted that CAN files indicate a pedal activation only when the foot is strongly engaged. These detections provide data to fill the blanks of the CAN files. The SVM models were trained for each video to adapt the algorithm to the specificity of each driver (shoes, driving behaviour). It has to be mentioned that the SVM training is fast.
4. Four binary decisions are then obtained at each time stamps: the presence or not of a front passenger, the presence or not of a foot on each of the three pedals.



Crop a Region of Interest
Compute feature image descriptor HoG
SVM model pre-trained with labeled videos
Apply a machine learning algorithm called SVM
SVM model pre-trained with CAN files
Decision
Passenger presence
Yes
No
Foot on pedal
Yes
No

Fig. 3: *Framework of the HoG + SVM algorithm.*

1.2. Deep learning algorithms

Deep learning based algorithms were used for “Texting”, “Presence of hands on the wheel” and “Phone-to-the-ear conversation” detections. Deep learning is a branch of machine learning that tries to mimic the learning mode of the human brain by modeling high-level abstractions in a set of data. These methods have drawn attention these past years in the computer vision research community by surpassing the performances of traditional machine learning algorithms, such as Support Vector Machine or Random Forest, in the fields of object detection or classification. However, these methods have strong drawbacks, such as the need of a heavy architecture (use of GPU) as well as large training databases.

Numerous deep learning models have been developed for object detection, such as NoC, (fast/faster) RCNN, DeepBox, segDeepM, Yolo darknet, TensorFlow and so on [3]. These models are built as a hierarchy of neural networks. To adapt the model for a specific task, the weights of the networks need to be trained with positive and negative training samples. For example if we want to build a tree detector, the model must be fed with a high amount of images, associated with a text file providing the tree positions in each image containing one, to learn how trees look like.

We have chosen to use the TensorFlow [6] framework to address specific situations in driving videos such as monitoring the number of hands on the wheel, detecting the use of a phone-to-the-ear and tracking down texting situations. TensorFlow is a simple open source framework for object detection, developed by Google to train neural networks. It is based on a robust deep learning model called “GoogLeNet-OverFeat”.

Three deep learning models were trained, one for each kind of studied objects:

1. Hand on the steering wheel,
2. Hand holding a cellphone in texting situations,
3. Hand holding a cellphone to the ear in calling situations.

For each model, training samples were needed. Numerous frames, containing or not the studied objects, were manually labelled from several videos by defining bounding boxes specifying the studied objects localization. The TensorFlow framework is then trained for each of the three studied objects. Table 2 provides information about the training samples, such as the number of frames and videos used to train each model as well as the obtained convergence iteration. Figure 4 shows examples of annotated frames.

When trained, the models are tested on videos (different from the training set). Then, the tool provides an output file stating for each frame the positions of detected objects, if any, with a corresponding confidence score. These raw outputs can be post-filtered to temporally smooth responses and thus delete small ranges of consecutive responses to keep only long and robust ones.

The results and performances of the presented algorithms are presented in the next section.

Table 2 Information about training samples for TensorFlow.

	Model iteration	**#videos used**	**#positive frames**	**#negative frames**	**#frames**
Hands on wheel	230 000	6	53 590	6 623	60 213
Texting	50 000	23	2 238	9 878	12 115
Phone-to-the-ear	30 000	69	27 170	4 546	31 716

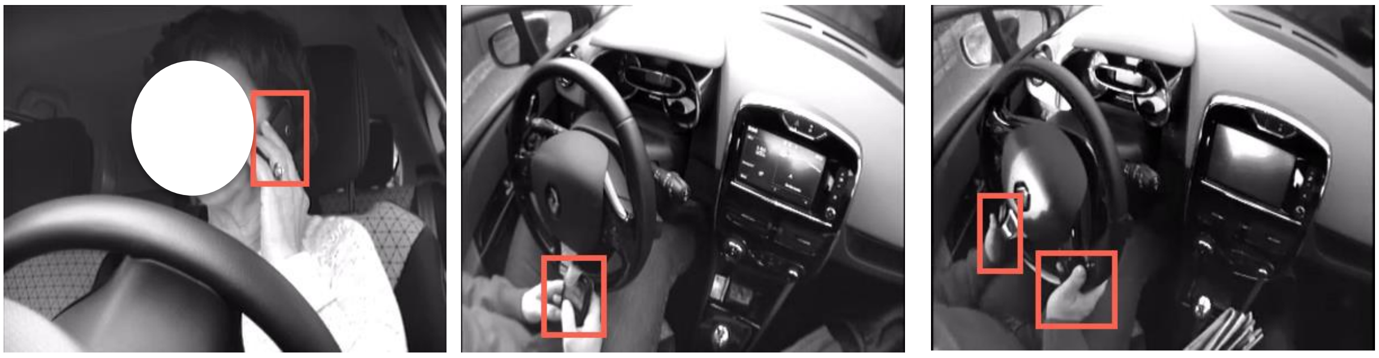


Fig. 4. *Examples of annotated frames (with bounding boxes in red) for the following tasks: cellphone-to-the-ear, texting, hands on the steering wheel*

2. Performances evaluation

The performances are presented for each secondary task. An evaluation protocol is first introduced, followed by the testing dataset used for each feature. Finally results are explained. It has to be mentioned that our solution could not be compared to state of the art methods because of the lack of shared public data and the differences between datasets.

2.1. Evaluation protocol

2.1.1. Testing datasets

Testing sets used to evaluate automatic detection are summarized in Table 3. Videos sample include various conditions (day and night videos, with several drivers and passengers, hands with or without gloves, different hand gestures and illumination etc). Figure 5 illustrates the diversity of the testing database for the different cameras.

To test and evaluate algorithms, positive and negative sequences are annotated (containing or not the studied secondary task). For example, if someone is texting from frame 1 to 100, this is labeled as a positive sequence. If the driver isn't texting from frame 101 to frame 500, this is a negative sequence. This labellisation is called a Ground Truth (GT). The proposed algorithm are then applied on the different testing videos and a binary result is obtained at each frame (for example presence of a passenger, use of a phone-to-the-ear). Two types of evaluations are used to assess the algorithms: frame-by-frame evaluation and sequence evaluation.

Table 3 Testing sets used to evaluate each detection: number of videos used and total duration in minutes. A balanced set means there is on average the same number of positive and negative samples. The testing sets are not the same for each item, even if some videos can be found in several sets.

Detection	#videos	Total duration (in minutes)	Balanced set
Passenger presence	21	51	Yes
Foot on top of pedal	4	20	Yes
Hands on the steering wheel	6	80	89% of frames with hands
Phone-to-the-ear	11	175	7.5% of frames with phone
Texting	6	42	10% of frames with phone

2.1.2. *Evaluation frame-by-frame*

After running the algorithms on the testing videos, the results are compared frame-by-frame with the labels. The following metrics are computed (see Table 4):

- TP: number of True Positives (for example the number of frames detected and labeled as "texting").
- FP: number of False Positives (for example the number of frames detected as "texting" while labeled as "not texting").
- TN: number of True Negatives (for example the number of frames detected and labeled as "not texting").
- FN: number of False Negatives (for example the number of frames detected as "not texting" while labeled as "texting").
- Precision = TP/(TP+FP), gives an information about the false positive rate among all the detections.
- Recall = TP/(TP+FN), gives an information about the capacity of the algorithm not to miss detections.

  These frame-by-frame metrics, currently used in classification methods [21], give an overview of the performance of the algorithm. To be used in driving studies analysis, sequences are needed in particular for rare events such as texting or phone-to-the-ear.

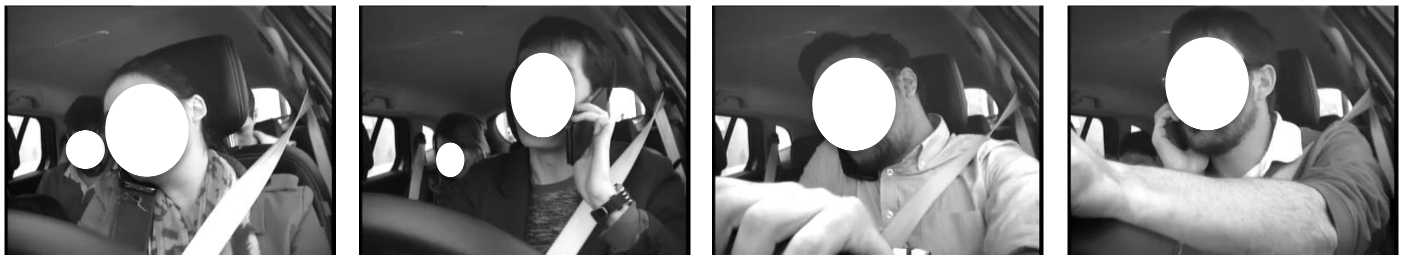
Table 4: Performance indicator definition.

		Algorithm predictions	
		+ (secondary task)	- (no secondary task)
Ground Turth (GT)	+ (secondary task)	TP	FN
	- (no secondary task)	FP	TN

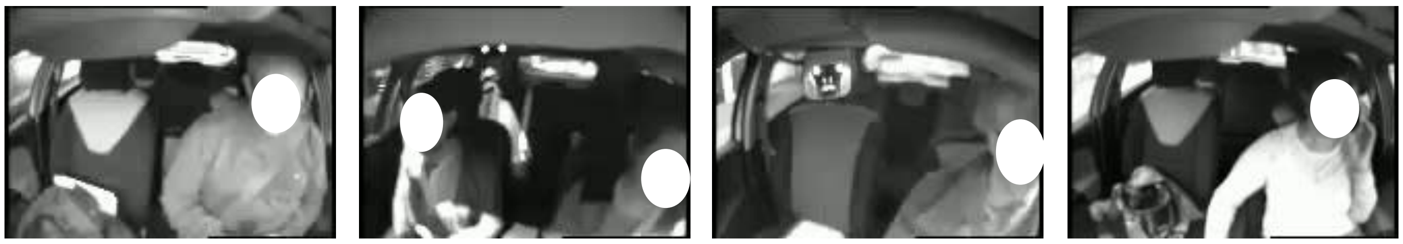
2.1.3. *Evaluation by sequence*

This type of evaluation consists in considering events as a sequence, by grouping all consecutive detections, rather than looking at the percentage of frames correctly classified. While the previous type of evaluation is used for all detections, this one is used only for the "texting" and "phone-to-the-ear" detections It allows to check if sequences with phone use, that don't happen frequently in real conditions, have a high detection rate and an "acceptable" false detection rate.

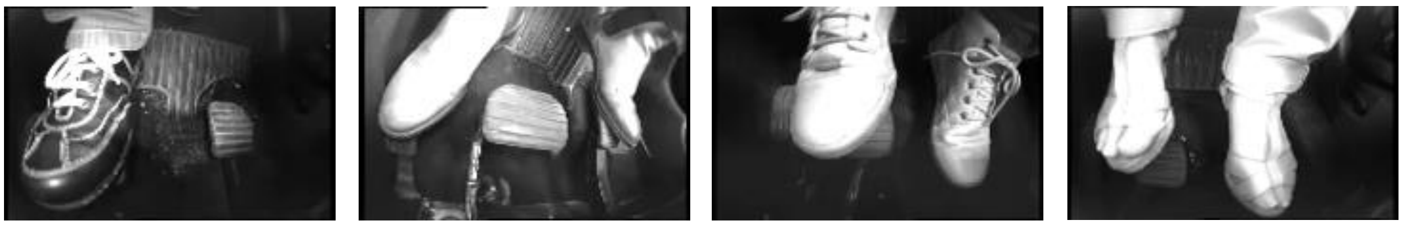
a



b



c



d

Fig. 5. *Examples of frames from the different testing sets illustrating the diversity of the test database. Frames are extracted from top view (a), driver (b), cockpit (c) and pedals (d) videos.*

We compute then three metrics for each videos:

- True Positive Sequence (TPS): for each GT sequence containing a phone, overlap between this sequence and all the detected sequences is computed. If the number of overlapped frames is higher than a threshold (here 30%), the GT sequence is considered as correctly detected.
- False Negative Sequences (FNS): All the sequences from the GT that are not validated (meaning that less than 30% of the frames have been detected as “phone”).
- False Positive Sequence (FPS): All the detected sequences that have no overlaps with any GT sequence.

2.2. Results

2.2.1. Passenger detection

For this detection, the HoG+SVM algorithm presented at the section 4.1 is used. **Precision rate is 95.6% and recall is 99.8%.** This algorithm is thus able to correctly detect almost all the frames where a passenger is present. In the meantime, 4.4% of the frames are wrongly classified as “passenger presence”. These few mismatched frames turned out to be complicated situations, like high saturation, shadows, presence of a bag or low luminosity (examples can be found in Figure 6). However, since a passenger cannot suddenly disappear from the seat while the car is moving, these errors can be deleted by smoothing the results on long sequences.

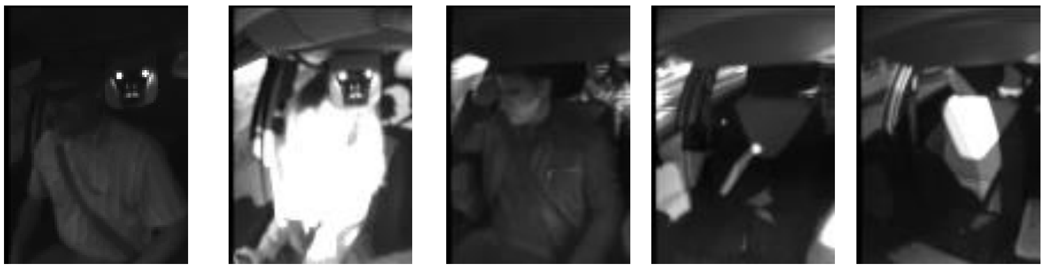


Fig. 6. *Examples of frames wrongly classified.*

2.2.2. Foot on pedal detection

HoG+SVM algorithm is used again for this detection. Transition phase (shifting of a foot from one pedal to another) are not labelled and will not be studied. Table 5 gives performances for the three pedals. **The precision rate is close or equal to 100%** for each pedal, and the **recall is between 94% to 98%.** Unlike the “passenger presence” detection, this means that the algorithm rarely provides a wrong output. For example, when a foot on the gas pedal is detected, it is nearly sure that there is indeed a foot there. On the other hand, when there is actually no foot on a pedal, the classifier rarely finds one. However, a lower recall points out that a few frames with the presence of a foot are missed.

Table 5 Performances of the HoG+SVM algorithm for the “foot over a pedal” detection. Results are given for each pedal (gas, brake and clutch).

	Gas pedal	**Brake pedal**	**Clutch pedal**

Precision	99.4%	100%	99.9%
Recall	98.0%	94,0%	98,2 %

2.2.3. Hands on the steering wheel detection

The “hands on wheel” detection is based on the deep learning algorithm presented in Section 4.2. The “hand on wheel” detector obtained with the TensorFlow framework is applied at each frame of the “from top” videos. We focus on detecting when the driver has at least one hand on the steering wheel. In the end, result is a binary output: no hand on the wheel/at least one hand on the wheel. Results are presented in Table 6. **The precision rate is 99,5%, and the recall is 85,4%.** This means that the algorithm is nearly never wrong when it indicates that there is a hand on the steering wheel. However, around 15% of frames labeled as “hand” are missed. A precise filter of the outputs could be used to fill the detection blanks and thus increase the recall rate.

Table 6: Confusion matrix for the hands on wheel module frame-by-frame (raw outputs).

Label	**At least one hand**	**No hand**
At least one hand detected	TP = 45642	FP = 215
No hand detected	FN = 7802	TN = 6694

2.2.4. Texting detection

The “texting” detection is based on the deep learning algorithm presented in Section 4.2. The phone detector obtained with the Tensorflow framework is thus applied at each frame of the “from top” videos.

Table 7 presents the confusion matrix of the results. The precision rate here is **21%** and the recall **47%**. This means that half of the frames labeled as “texting” are correctly detected, but 4 out of 5 detections are wrong. However, since the testing set is not balanced, there are definitely more frames labeled as “no phone use” (92.5%). This is due to the fact that these events are rare. We can interpret these numbers differently, like the percentage of frames correctly detected for each label. For the label “texting”, this is equal to the precision rate thus 47%. But for the label “no phone use”, this corresponds to 16%. So even, if 16% of frames are wrongly labeled, they still correspond in the end to 79% of the wrong outputs since they are more present.

In the evaluation by sequence, 22 out of 32 (68%) GT sequences of “texting” are correctly validated. In all the detected sequences, 3186 out of 3336 (95%) are considered as false alarms (FPS). This high number can however be explained by observing the histogram of the FPS lengths (Figure 7). Majority of these false alarms are very short (less than 10 frames).

Table 7 Confusion matrix for the texting module frame-by-frame (raw outputs). The last two columns present the results in percentage for each label.

Label	**Phone use**	**No phone use**	**Phone use (%)**	**No phone use (%)**
Texting	TP = 4667	FP = 17690	TP = 47%	FP = 16%
No phone detected	FN = 5248	TN = 95375	FN = 53%	TN = 84%

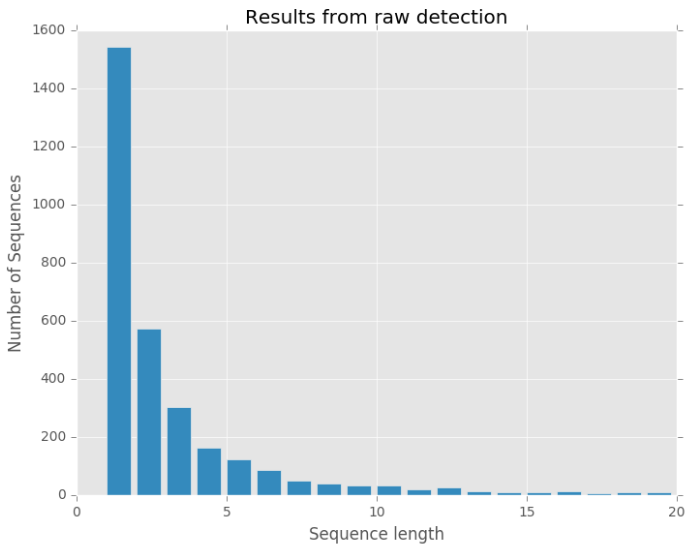


Fig. 7. *Histogram of the lengths of false positive sequences (FPS) for the texting module (raw outputs).*

To filter these raw outputs, we use morphological operations (opening operation to delete small responses and closing operation to merge range of responses). We obtain a better number of TP and thus a better **recall rate: 67%** (Table 8). In term of sequences, 25 GT sequences out of 32 are validated (78%), and 352 FPS are obtained out of 381 (92%). While the number of FP is indeed increased (resulting in a lower **precision rate of 17%**), FPS was strongly reduced, going from 3336 to 381, and the number of TPS has increased, going from 22 to 25. Also, most of the false positive sequences are very short (Figure 8).

Table 8 Confusion matrix for the texting module frame-by-frame (filtered outputs). The last two columns present the results in percentage for each label.

Label	Phone use	No phone use	Phone use (%)	No phone use (%)
Detected phone	TP = 6620	FP = 33324	TP = 67%	FP = 29%
No phone detected	FN = 3295	TN = 79741	FN = 23%	TN = 71%

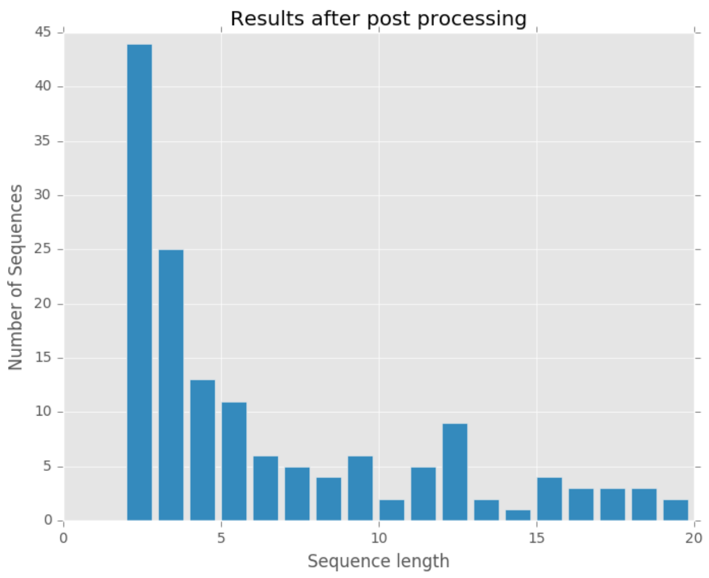


Fig. 8. *Histogram of the lengths of false positive sequences (FPS) for the texting module - filtered outputs.*

2.2.5. Phone-to-the-ear detection

The "phone-to-the-ear" detection is based on the deep learning algorithm presented in Section 4.2. As for the previous item, the **precision (18%) and recall (50%)** rates are pretty low (Table 9). In term of sequences, 10 out of 14 (71%) sequences are correctly validated into the GT, and 924 out of 1030 (89%) of FPS. As for the "texting" detection, most of these false detected sequences are shorter than 10 frames (Figure 9).

With a morphological filter (Table 10), a **better recall rate (67%) and a lower precision rate (13%)** are obtained. 12 out of 14 (86%) sequences from the GT are correctly validated and 103 out of 115 (89%) of FPS. Like the "texting" detection, the increase of the number of FP is misleading. Indeed, the number of FPS has been divided by ten, while the number of TPS has increased from 10 to 12.

Table 9 Confusion matrix for the phone-to-the-ear module frame-by-frame (raw outputs). The last two columns present the results in percentage for each label.

Outputs	**Phone use**	**No phone use**	**Phone use (%)**	**No phone use (%)**
Detected phone	TP = 1541	FP = 7255	TP = 50%	FP = 26%
No phone detected	FN = 1530	TN = 20897	FN = 50%	TN = 74%

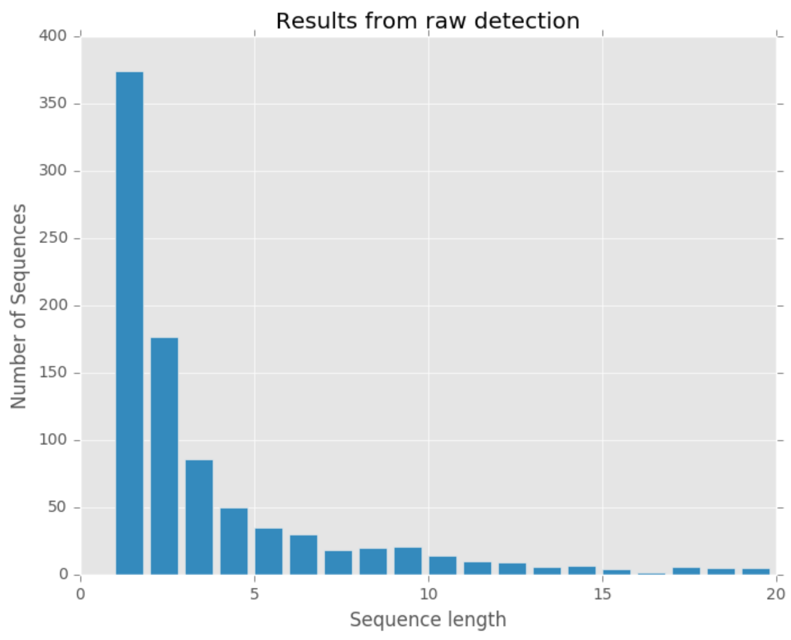


Fig. 9. *Histogram of the lengths of false positive sequences (FPS) for the phone-to-the-ear module (raw outputs).*

Table 10 Confusion matrix for the phone-to-the-ear module frame-by-frame (filtered outputs). The last two columns present the results in percentage for each label.

Outputs	**Phone use**	**No phone use**	**Phone use (%)**	**No phone use (%)**
Detected phone	TP = 2045	FP = 14020	TP = 67%	FP = 50%
No phone detected	FN = 1026	TN = 14132	FN = 23%	TN = 50%

3. Perspectives and conclusions

Results of this study are promising, it is indeed possible to detect secondary tasks with computer vision algorithms. However, there is still the need to lower the number of false detections. Actually, we aimed to maximize the number of true positives even if that meant having a high number of false positives. Automatic detections are still really helpful for NDS, since it allows to strongly reduce the time to compute manual annotations.

Several improvements should be considered for future works. Post-processing filtering of TensorFlow outputs can be improved to remove false positive situations. The Long Short-Term Memory networks (LSTM) [19] included in TensorFlow software takes into account the temporal information during a video, similar as a tracking step: it is able to learn long-term dependencies. It could be helpful to learn a more precise model, as well as adding more training data. Indeed for deep learning approaches, the more data we have, the more efficient deep learning model we can obtain. A thorough comparison with state of the art algorithms should be helpful as well.

Speech detection could also be a useful tool for secondary tasks analysis, especially to detect situation of conversation using Bluetooth function or hands free headset. The association with the "passenger presence" information also allows to assess the situation "talking with a passenger/not talking". We have already tested several algorithms based on face landmarks and mouth opening but there were not successful. Driver's movements and the challenges presented by the dataset (images in gray level, occlusions due to the wheel, camera angle…) made difficult the landmark detection. Therefore they were not precise enough to perform reliable detection.

The LAB intends to test this first version of automatic detection of secondary tasks on Udrive data. In this European naturalistic driving study [17], approximately 25 000 hours of driving in 5 different countries will be available. The automatic processing of a set of video data should provide very interesting variables that analysts will use to study secondary tasks linked with nomadic devices.

4. Acknowledgments

We would like to thanks the LAB PSA-Renault, in Nanterre (France) for the funding.

5. References

5.1. Websites

[1] Kaggle Competitions: https://www.kaggle.com/c/state-farm-distracted-driver-detection

[2] Dollar P., Piotr's Computer Vision Matlab Toolbox (PMT): http://pdollar.github.io/toolbox/

[3] References of numerous deep learning algorithms: https://handong1587.github.io/deep_learning/2015/10/09/object-detection.html

[4] The second Strategic Highway Research Program (SHRP-2) Dataset:
http://www.trb.org/StrategicHighwayResearchProgram2SHRP2/Blank2.aspx

[5] Vision for Intelligent Vehicles and Applications (VIVA) Dataset: http://cvrr.ucsd.edu/vivachallenge/

5.2. Journal articles

[6] Abadi M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, *Journal CoRR*, 2016.

[7] Ohn-Bar E. et al. Driver hand activity analysis in naturalistic driving studies: challenges, algorithms, and experimental studies. *J. Electronic Imaging*, 2013.

[8] Viola P. and Jone M. J.. Robust Real-Time Face Detection. *Int. J. Comput. Vision 57*, 2004.

5.3. Conference paper

[9] Cortes C. and Vapnik V.. 1995. Support-Vector Networks. *Machine Learninig, 20, 3* (September 1995), 273-297.

[10] Dalal et al. 2005. Histograms of Oriented Gradients for Human Detection. *InProceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01 (CVPR '05), Vol. 1. IEEE Computer Society, Washington, DC, USA*, 886-893.

[11] Das N. et al. On Performance Evaluation of Driver Hand Detection Algorithms: Challenges, Dataset, and Metrics, *2015 IEEE 18th International Conference on Intelligent Transportation Systems.*

[12] Hoang Ngan Le T. et al. "Multiple Scale Faster-RCNN Approach to Driver's Cell-phone Usage and Hands on Steering Wheel Detection", *Computer Vision in Vehicle Technology, CVPRW, Las Vegas, July, 2016.*

[13] Ohn-Bar E. et al. Head, Eye, and Hand Patterns for Driver Activity Recognition, *2014 International Conference on Pattern Recognition.*

[14] Rangesh A. et al. Long-term, Multi-Cue Tracking of Hands in Vehicles, *IEEE Transactions on Intelligent Transportation Systems, 2016.*

[15] Seshadriv K. et al. Driver cell phone usage detection on Strategic Highway Research Program (SHRP2) face view videos, CVPR 2015, *IEEE Conference on Computer Vision and Pattern Recognition.*

[16] Tran C. et al. Modeling and prediction of driver behavior by foot gesture analysis. *Comput. Vis. Image Underst, 2012.*

[17] Van Nes N. et al. Issues and Barriers Faced when Designing and Running the European Naturalistic Driving Study UDRIVE, *VVTI 5th symposium on NDS 2016.*

[18] Xiong X. and De la Torre F., Supervised Descent Method and its Application to Face Alignment), *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013*

5.4. Book, book chapter and manual.

[19] Hochreiter S. and Schmidhuber J. 1997. Long Short-Term Memory. *Neural Comput. 9, 8* (November 1997), 1735-1780.

5.5. Report

[20] "Driving performance assessement- methods and metrics", AIDE project, Deliverable D2.2.5-2004.

[21] Davis J. and Goadrich M., The relationship between Precision-Recall and ROC curves, *In Proceedings of the 23rd international conference on Machine learning (ICML), 2006*