Detection of Distractions of Drivers

Anders Austlid Tasken and Marius Hernes Brateng
University of Trento, Italy

I. Abstract

On a world basis, over 1 million people die in traffic accidents each year [1]. That makes up approximately over 3000 deaths on the road each day. A great part of car accidents are caused by distraction of drivers. Usage of smart phones and interaction with passengers are two of many reasons which can cause such distractions. In order to reduce the amount of accidents in the traffic, it is possible to build a system capable of detecting inattention among drivers. With the use of state of the art computer vision techniques, drivers that do not pay attention can be detected and warned. Throughout this report, an implementation of this system is discussed, implemented and evaluated.

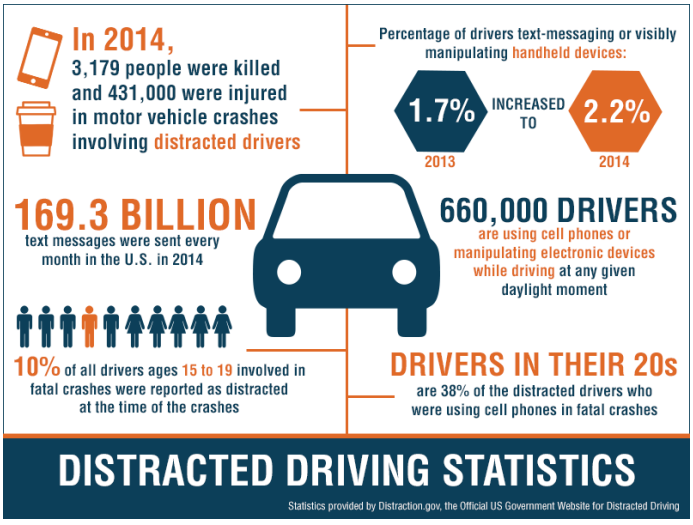

Fig. 1: Causes of crashes from [6]. Statistics provided by Distraction.gov, the official US Government website for Distracted Driving

II. Introduction

Detection of inattention among drivers is a problem that might be solved with computer vision due to the huge improvements of the technology in the field in recent years. With the use of object detection and face mapping algorithms it is possible to determine, with some certainty, if the driver is paying attention to the road or not. This report mainly focuses on detecting drivers that do not look at the road due to falling asleep, talking with passengers etc. The report also focuses on detecting objects near the driver usually connected to inattention, among them cellphones and similar objects.

III. Theory

A set of essential theoretical concepts and algorithms related to the problem statement is presented below.

A. Object detection

Object detection is a technology within the field of computer vision where the goal is to detect semantic objects in a digital image or video. Each object is characterized by its unique set of features. A feature is defined as specific structures in the image such as points or edges.

Both the runtime and the accuracy of different object detection algorithms is varying. One of the faster implementations of object detection was first introduced in 2015 in the paper "You Only Look Once: Unified, Real-Time Object Detection" [3]. This implementation is named You Only Look Once (YOLO), and is a single-stage detector which handle the object detection as a regression problem. The widespread use of YOLO is due to the fact that it is fast enough to run on real time applications, and can be used for live surveillance. See figure 2 for a visualization of the technique.

Single-stage detectors learn bounding box coordinated and class label probabilities simultaneously. Typically, these kind of detectors performs much faster than two-stage detectors, but they are less accurate as well.

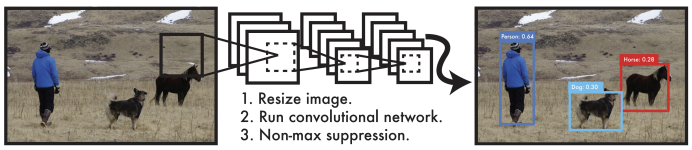

Fig. 2: Yolo object detection methodology from [3].

B. Face detection

Face detection is an important aspect of computer vision, and there exists several well working implementations that produce good results. The main purpose is to be able to identify human faces in digital images. Face detection can be regarded as a special case of object detection with the specific case of detecting faces, and not the general situation of detecting objects.

One algorithm used for face detection take the use of Histogram of Oriented Gradients (HOG) and a linear classifier. HOG is a feature descriptor which extract the local shape and appearance of an object by computing the distribution, or histogram, of intensity gradients and edge directions. This feature descriptor can be used as input in a Linear Support Vector Machine (linear SVM), which is a linear classifier. By training the linear SVM on labeled data, the classifier is able to separate and classify faces with good performance. By using a sliding window detection scheme, faces located in different places in the image can be detected. With the use of an image pyramid, in other words scaling the image to

different resolutions, faces of different sizes can be detected as well.

C. Face mapping

Facial mapping is a technique that are able to extract information about the face. The position of landmarks in the face is found, where a landmark is a point in the face with a specific set of characteristics. The face mapping perform, in other words, a localization of key points within the face. The facial landmarks can be used to localize and analyze different facial regions, such as the eyes, mouth, nose or eyebrows.

A state-of-the-art implementation of facial mapping that performs face alignment in milliseconds, is presented in the paper named "One Millisecond Face Alignment with an Ensemble of Regression Trees" [2]. This implementation use a streamlined cascade structure of regression functions to obtain high speed without a loss of accuracy. The algorithm is able to detect 194 facial landmarks on a face from one single image in a millisecond. The landmarks obtained is shown in an example in figure 3.

Fig. 3: Face mapping example from [2].

D. Closed eyes detection

By taking the face mapping landmarks as input, closed eyes is detected with the methodology described in "Eye blink detection with OpenCV, Python, and dlib"[10]. With the use of the 6 coordinates that represent the eye from the facial landmarks, it is possible to compute the relation between the width and height of the eye, called the eye aspect ratio (EAR). The equation describing this relation was presented by Soukupová and Čech in a paper named "Real-Time Eye Blink Detection using Facial Landmarks" [11], and is given in equation 1.

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\,\|p_1 - p_4\|} \tag{1}$$

where $p_i, i \in [1, 6]$ is 6 points in the eye given from figure 4, as presented in [11].

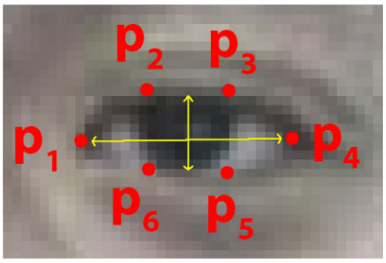

Fig. 4: Points defining an eye from [11].

IV. Solution

The implementation of the driver attention detection system is obtained with the use of a stereo camera, an object detection algorithm, a face detection algorithm and a face mapping algorithm. By combining these techniques, and using the data retrieved from the stereo camera as input, situations indicating inattention is detected. By analysing the data received from the face mapping algorithm, closed eyes and looking away is detected. The YOLO object detection is used to detect cell phones. A more detailed explanation of the solution is placed below.

A. Stereo camera

To retrieve information about the driver, a stereo camera is used to maintain a video with depth information. The camera used is provided by StereoLabs, and is named ZED [4]. A stereo camera is used to produce better results by only evaluating the driver within the image plane. The driver is assumed to be a given distance in depth from the camera, and thus can be distinguished from passengers in the back seat.

B. YOLO object detection

To detect the use of a cellphone while driving, the YOLO object detection algorithm was used [3]. The technique is presented and described in section III-A. The implementation of the algorithm is based on the code presented in the publication by Adrian Rosebrock [5].

The algorithm use a library named Open source computer vision (OpenCV) to load the pre-trained YOLO object detector, trained on the COCO dataset. OpenCV is also used to load the video stream, construct a blob from the input frame, suppress weak, overlapping bounding boxes and write bounding boxes on top of the video in an .AVI file.

To be able to maintain fast runtime, it is needed to do the computations on the graphics processing unit (GPU). To perform general purpose processing on the GPU, the application programming interface (API) model created by Nvidia named CUDA is used. CUDA is used on certain computations performed by the OpenCV library, resulting in significant improvements in runtime.

With the use of YOLO object detection, OpenCV and CUDA, objects in a video stream is detected and marked by a bounding box. This information is later used to detect drivers that do not pay enough attention to the driving.

C. Dlib face detection

In order to detect faces in the video stream, a face detector provided by the dlib library is used [7]. The methodology used in the face detector is similar to the description presented in section IV-C. The output of the face detection is used as input to the facial mapping algorithm, in order to retrieve further information about the face. This is necessary to evaluate if the driver is not looking at the road, sleeping etc. The output is an array of the faces detected by the algorithm.

D. Dlib face mapping

By looping over the faces given from the face detection algorithm, it is possible to find facial landmarks on all detected faces in the video stream. The dlib library [8] is used to perform the extraction on facial landmarks, as described in section IV-D. A library named "imutils" [9] is used to convert the facial mappings to a numpy array. The facial landmarks is marked on the video stream as points on the given location.

E. Closed eyes detection

The relation between the width and height of the eye (EAR), described in section IV-E, is computed with the facial landmarks as input. If the computed EAR value is below a specified threshold for a given frame, the eye is considered to be closed. This threshold value is found to produce best results if it is equal to 0.3. If the EAR value is below the threshold for a given number of subsequent frames, the driver is evaluated to be sleeping. This detected action is marked with a box in the video frame, as seen in figure 5. The threshold value for number of subsequent frames with eyes closed is set in seconds, then converted to number of frames by the frames per second ratio (FPS) which is given from hardware used. The value used during testing is 2 seconds, but further research could be done to make the value more appropriate.

F. Looking away detection

The event where the driver is looking in the wrong direction, i.e. not forward on the road, is detected with the use of facial landmarks. The dlib face mapping is not able to find facial landmarks if the direction of the face give too big angle relative to the camera. Thus, every time the dlib face mapping fail to extract landmarks, it is an indication on a person looking in another direction. If the number of subsequent frames dlib fails to extract landmarks is above a threshold value, the driver is considered to not pay enough attention to the road. This is marked as shown in figure 6. The threshold value for number of subsequent frames with the driver looking away is set in seconds, and is set to 2 seconds, similar to the case of closed eyes. This threshold value can easily be changed. The figure 7 show a detection of looking away for under 2 seconds, and hence no warning is shown.

G. Cell phone detection

The use of a cell phone while driving is also considered as a safety violation, and is of that reason detected and marked in the algorithm. If the YOLO object detection is able to detect a cell phone in the video stream with a certainty above a threshold, the number of subsequent frames with the phone visible is counted. If the count exceed a threshold, the algorithm mark the use of cell phone with a box in the frame, presented in figure 8.

V. Evaluation

The final result of detecting inattention and safety violations while driving is given as a video stream output. In the video, the three actions described in section IV-E, IV-F and IV-G is marked with text on top of the input video as showed in figures 8.

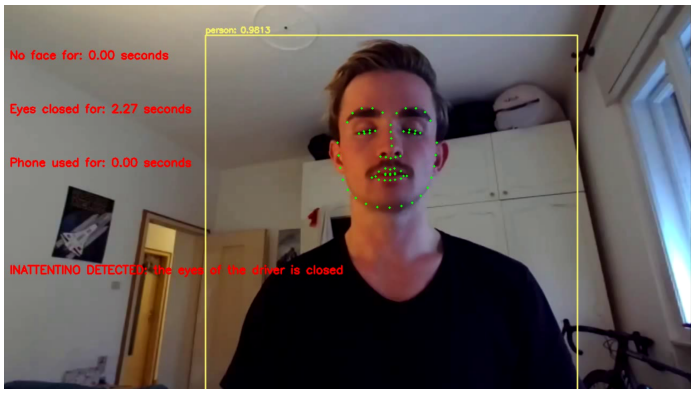

Fig. 5: A screenshot of a test video where closed eyes is detected, and is above the threshold value of 2 seconds.

Fig. 6: A screenshot of a test video where sleeping (looking away) is detected, and is above the threshold value of 2 seconds.

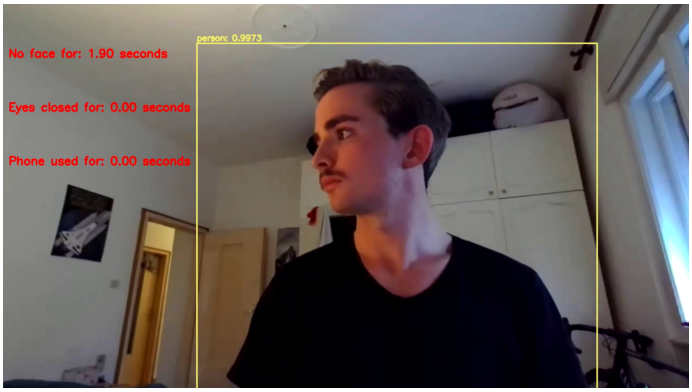

Fig. 7: A screenshot of a test video where looking away is detected, and is below the threshold value of 2 seconds and thus not marked.

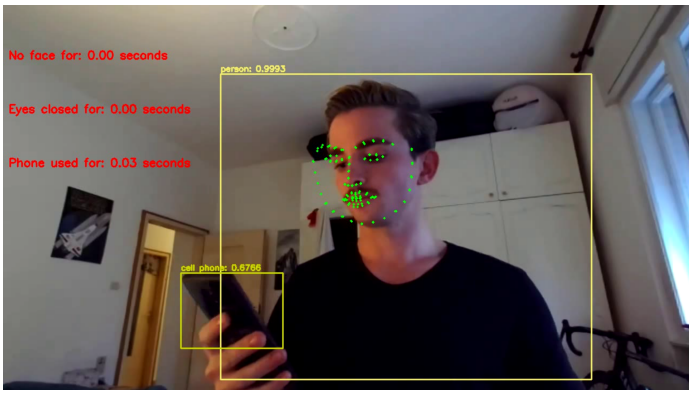

Fig. 8: A screenshot of a test video where a cell phone is detected, and is below the threshold value of 2 seconds.

The performance of the algorithm is stable with regard to detection of closed eyes and looking away. The detection of the face, and mapping of landmarks, is able to produce good results in varying settings. These algorithms are not very dependent on constant illumination settings nor slow movement. They are however dependent on a video input with a face close to the camera. Since these two algorithms give stable predictions, both the closed eye detection, from section IV-E, and the looking away detection, from section IV-F, have good performance in the tested environment.

The YOLO object detection algorithm do not detect the cell phone in all frames where a phone is used. This detection is more prone to illumination, movement, angle of the object and size of the object. Thus, the number of subsequent frames with a detection of a cell phone rarely exceeds the threshold value.

The efficiency of the algorithm is dependent on the runtime of the YOLO object detection algorithm, the face detection algorithm and the face mapping algorithm. The rest of the processing is relatively irrelevant, and can be discarded in the efficiency evaluation. The efficiency of the algorithms is tested separately, which give a significant difference in runtime between the detection algorithms. The face detection and face mapping of landmarks is able to produce stable results in real time. The YOLO object detection is a more complex detection algorithm, and is dependent on a lower resolution input video, fewer frames per second video and computations completed on the GPU. YOLO is able to perform real-time results with the correct setup, and is much faster than most of the competing detection algorithms. Without performing general purpose processing on the GPU, the algorithm is not able to give real-time results.

The detected events in the implementation is dependent on the right usage to prevent traffic accidents. Drivers should only be notified if there is a high probability of inattention, and thus some of the detected events should be filtered out. The notification of the driver should be discreet in order to reduce the risk of creating annoyance.

VI. Improvements

Even though the performance of the algorithm give usable results, there is room for improvement. First and foremost it is useful to make the algorithm able to run in real time. This is necessary to be able to use the implementation in a real world scenario to prevent traffic accidents. The given implementation take a pre-recorded video stream as an input, but it is possible to take a running video stream as input and detect inattention while recording.

The threshold of the number of subsequent frames before warning the driver is a trade off between false positives, false negatives and warning the driver as fast as possible. If the car is doing 100 km/h, it moves across the distance of a football pitch in three second. Hence, time is a great matter in this case. The threshold of the number of subsequent frames with eyes closed, looking away and a cell phone in the image should be set to a high value to reduce the number of false positive, to avoid irritating the driver. False negatives is not wished for either, since an accident might occur if the driver is not warned. On the other hand, it is really important that the driver is warned as fast as possible if distractions are detected. One could also cope with this problem with more advanced filtering. Correlation between two frames could be taken into account, or one could specialize a Neural Network to recognize sleeping persons for this particular usage.

The depth information from the stereo camera is not used in the final result, but should be used to separate the driver from passengers. For example, the case where the driver use a cell phone should be separated from the situation where the passenger uses one. Similarly, the detection of the face should only be evaluated if the depth is within the range of the driver seat.

The detection of a cell phone has potential to be improved as well. Since the YOLO object detection produce unstable predictions, it is possible to apply techniques to use the detection in an optimal manner. I.e. it is an option to track the number of frames a phone is detected during the last seconds, in stead of the previous subsequent frames.

VII. Conclusion

The implementation of a detection of inattention among drivers obtain results which might be used to notify drivers, and thus prevent potential accidents. With the use of the detection of safety violation events as a system to notify the

driver in a subtle manner, the number of traffic accidents may be reduced. There is room for improvements with regard to runtime and stability. All in all, the implementation might be a step towards safer road traffic.

REFERENCES

[1] The Association for Safe International Road Travel (ASIRT). *Annual Global Road Crash Statistics*. Opened 07.02.2020. [*https://www.asirt.org/safe-travel/road-safety-facts/*]

[2] Kazemi, Vahid. Sullivan, Josephine . *One Millisecond Face Alignment with an Ensemble of Regression Trees*. KTH, Royal Institute of Technology. [*https://www.researchgate.net/publication/264419855*] June 2014, Conference Paper. Conference: Computer Vision and Pattern Recognition, At Columbus, Ohio, USA.

[3] Redmon, Joseph. Divvala, Santosh. Girshick, Ross. Farhadi, Ali. *You Only Look Once: Unified, Real-Time Object Detection*. University of Washington, Allen Institute for AI, Facebook AI Research. [*https://arxiv.org/abs/1506.02640*] Submitted on 8 Jun 2015 (v1), last revised 9 May 2016 (this version, v5).

[4] Stereolabs Inc. *ZED camera*. Stereolabs Inc, San Francisco, CA 94107. [*https://www.stereolabs.com/zed/*]

[5] Rosebrock, Adrian. *YOLO object detection with OpenCV*. Deep Learning, Object Detection, Tutorials. [*https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-OpenCV/*] November 12, 2018.

[6] Weiss, Anapol. *Distracted Driving Facts and Statistics*. Opened 07.02.2020. [*https://www.anapolweiss.com/distracted-driving-facts-and-statistics/*] May 4, 2016.

[7] King, Davis E.. *Dlib frontal face detector*. Dlib c++ library. [*http://dlib.net/dlib/image_processing/frontal_face_detector_abstract.h.html*] Copyright (C) 2013.

[8] José, Italo. *Facial landmarks recognition*. [*https://github.com/italojs/facial-landmarks-recognition-*] Jun 29, 2018.

[9] Rosebrock, Adrian. *imutils*. [*https://github.com/jrosebr1/imutils*] Accessed 08.12.2019.

[10] Rosebrock, Adrian. *Eye blink detection with OpenCV, Python, and dlib*. [*https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/*] Accessed 07.02.2020.

[11] Soukupova, Tereza. Cech, Jan. *Real-Time Eye Blink Detection using Facial Landmarks*. Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague. [*http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf*] February 3–5, 2016.