sciendo

Strojnícky časopis – *Journal of* MECHANICAL ENGINEERING,
VOL 74 (2024), NO 3, 55 - 68

STU SjF

A LOW-COST VEHICLE ASSISTANCE SYSTEM FOR DETECTION OF CRITICAL DRIVING SITUATIONS

HLAVATÝ Michal[1*], CIGÁNEK Ján[1], KOZÁKOVÁ Alena[1], KUČERA Erik[1], HAFFNER Oto[1]

[1] Slovak University of Technology in Bratislava, Faculty of Electrical Engineering and Information Technology, 841 04 Bratislava, Slovakia, e – mail: michal_hlavaty@stuba.sk

Abstract: The paper presents a low-cost driver fatigue detection camera system for older car models that do not have built-in fatigue detection systems. Based on a review of current solutions and image processing methods for face and eye detection, the Viola-Jones method for face detection and the cascade regression for eye detection were selected to enable real-time detection via smartphones and avoid the high cost of traditional systems. The proposed system has been successfully tested with real-time video under laboratory conditions.

KEYWORDS: assistance system; fatigue detection; face detection; eye detection; image recognition

1 Introduction

Due to technological advancement, road traffic is becoming safer in the last years. Still, many people are killed in road accidents every year. According to the National Highway Traffic Safety Administration (NHTSA), 697 people died in the U.S. in 2019 due to fatigued driving. In Germany, drowsiness behind the wheel is the cause of 25% of all freeway traffic accidents [1]. While fatigue detection systems are nearly always installed in newer cars, they may not be in older models. Costs of installing these kinds of assistance systems might be rather significant. The article reviews current systems for detecting driver fatigue, examines various approaches that can be used to identify fatigue, and gives a brief summary of image processing techniques which can be used to identify the driver's fatigue from theirs face and eyes.

To better understand fatigue driving, we need to define the terms drowsiness and fatigue first. Considering their differences, both the words fatigue "tiredness" and drowsiness "sleepiness" are sometimes used synonymously. Tiredness is defined as the inability to do a certain activity caused by mental or physical labor, or due to the repetition of the same action over an extended period. On the other hand, drowsiness is defined as a desire to sleep which is triggered by a strong biological drive [2]. In this paper we will mostly use the term “fatigue”.

The fatigue of the driver could be defined as a progressive decrease of awareness due to several subjective or environmental factors. The state of drowsiness leads to blurry vision and obscured consciousness, which is followed by sleep [1,2].

According to [3], the best way to prevent sleeping behind the steering wheel is to detect and prevent it in its early stages. There are many methods of drowsiness detection in the vehicle. Generally, the methods are classified in one of the following main areas: detection of physiological phenomena of the driver, detection of actions of a driver, detection of behavior of a driver, driver’s feedback, gathering data about driving style. Let us briefly introduce the most widely used fatigue detection methods.

DOI: 10.2478/scjme-2024-0033, Print ISSN 0039-2472, On-line ISSN 2450-5471

Fatigue detection based on EEG [4]. Electroencephalogram is a physiological signal that is measured to determine if the person is drowsy or awake. The signal has several bandwidths, each representing state of awareness of the person. If the measured frequency falls within the bandwidth 4-8 Hz (state of drowsiness) or a lower threshold of the bandwidth 8-13 Hz (state of relaxation), the system alerts the driver. This method is extremely precise; however, its huge disadvantage is a cumbersome device to measure the EEG signal.

Fatigue detection based on steering wheel movement [5]. This method detects the micro-reactions on steering wheel. In the awake state, the driver, always calculates the direction of the vehicle based on parameters of the road like its curvature, width, or the wear. Typically, the driver adjusts the ride with small, smooth movements of the steering wheel. When first signs of fatigue appear, the number of micro-reactions that adjust the flow of the ride starts to decrease. The fatigue detection system can detect this decrease and warn the driver; however, the method is too reliant on the properties of the road.

Fatigue detection based on yawning detection [6, 7]. Yawning is one of visual signs of fatigue deforming geometrical properties of the mouth. To detect those deformations, it is necessary to search for the face, and extract key features from its area. This system is straightforward but cannot be relied on by itself and has often to be combined with other face recognition systems.

Fatigue detection based on eye properties [8]. Other visual signs to detect fatigue are eyes closing and blinking frequency. Based on [9], the percentage of eyelid closure over the pupil over time (PERCLOS) is the most reliable and valid determination of a driver's alertness level. At any moment, one of the following states can be assigned to the eyes: fully opened, partially opened, closed. Based on the duration of individual eyes states, the system can detect level of fatigue of the driver. Contrary to the previous methods, this one is sufficient on its own and does not need any cumbersome device, however, can be improved with other synergistically cooperating systems.

Fatigue detection systems are often included in the extended car equipment, however, if not and the car owner to install it, the cost can be quite high, depending on the car manufacturer. For example, Škoda uses the fatigue detection system analysing the driver's behavior based on steering wheel to determine the level of fatigue. A warning tone is activated if the steering wheel's handling deviates from the normal, because the driver usually needs minor angular displacements in the wheel's alignment. The cost of such a system is 54 € without installation, which is one of lowest prices for such device [10]. Volkswagen also detects angular changes of the steering wheel, but it does not rely solely on it, rather employing additional sensors that monitor pedal usage and lane deviations. Multiple sensors fusion greatly increases reliability of fatigue detections and decreases false positives/negatives. The cost of this system depends on car model but ranges around 135 €, however this system also includes some additional features [11]. There are also manufacturers who create products vehicle independent models. Such devices work as stand-alone, modular add-ons. One example of such a device is iVue Driver Fatigue System made by RVS. This system uses artificial intelligence based on face and feature recognition, to accurately monitor the driver, detect any signs of fatigue from theirs face and eyes. The system also comes with a mountable camera, but price hovers around 350€ [12].

This article deals with the design of a simple low-cost fatigue detection assistant system based on face detection and facial features recognition. The article is organized as follows. In Section 2, principles and different approaches of face and eye detection are presented. Section 3 provides the main result of the article – a low-cost assistance system implemented first on a Raspberry PI 3 B+, and on a smartphone, because the Raspberry PI 3 B+ performance was not

sufficient to run the proposed algorithm in real-time. In the last section, obtained results are analysed and possible improvements are discussed

2 Overview of face and eyes detection methods

To build a functional low-cost fatigue detection device, which would not require an extensive, advanced installation and could be used by almost everyone we have chosen the fatigue detection from eyes. This approach works best even independently without the need of additional subsystems, and according to [8] and [6] provides quite precise results.

We have divided solution to this task into smaller subtasks. Firstly, we need to detect the face, this process needs to be fast enough, hence, the application has to run in real time. The area that holds most information about drowsiness on human face are the eyes, so next we need to focus on the eyes' area. In next step, we have to decide if the eyes are opened, and to which extent. The last subtask includes processing of the gathered data from video feed and deciding if the driver is fatigued.

Face detection can be carried out by an extensive amount of methods which can be divided into two main categories: feature-based and image-based. Feature-based methods detect faces in the image by extracting facial features like nose, mouth, or eyes subsequently using them to detect the human face. Image-based methods aim towards best matches between the testing and the training images [13, 14].

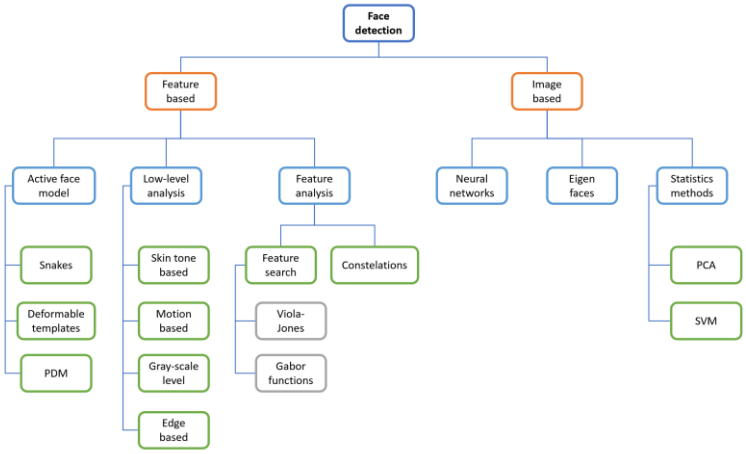

Fig. 1 Classification of face detection methods [13]

As mentioned before, we need to detect facial features, namely the eyes, thus the focus will be on feature-based methods. For example, the Edge-based detection, as the name suggests, aims to identify edges in the image, because irises or lips appear darker compared to nearby areas. This allows the algorithm to find local minimums. According to [15], this method can be both simple and fast enough. Another method considered was motion-based. It tries to detect motion of an object on the video recording and extract the facial features by simple image overlaying and gathering differences. According to the method proposed in [16], facial features are extracted in linear, three-layered iteration process. Both above-mentioned methods are quite fast and simple but can prove inefficient in non-optimal circumstances. Methods based on feature analysis are more complex and advanced but are more resilient and robust. This type of detection leverages facial knowledge and removes the ambiguity created by low-level analysis; it applies the sequential feature search, based on the relative positions of individual facial features. The major facial features are identified, which allows the algorithm to ignore the less prevalent ones. Using these methods, the image processing is fast

because it does not require learning [13, 14]. For example in [16], the reference point on the face are eyes and their position. When the algorithm detects the eyes in binarized images, for every pair of eyes it tries to find a nose, mouth, and eyebrows. The Viola-Jones method [17, 18] applied to detect faces in [16], is versatile enough to be used to find for any predefined object. The main advantage of the Viola-Jones algorithm is its robustness and possibility to detect objects in real time. To detect objects, this method uses Haar features, which is an area detector of characteristic features of given objects. It uses so-called Haar wavelets (flags); the values of a Haar wavelet e.g. defined by two rectangular areas, is calculated as the difference of the sum of the brightness values between the regions [14, 17]. Original collection of Haar features (Figure 2) was later expanded by flags which were rotated by 45° (Figure 3) which improved detections of the objects [17, 18].

Fig. 2 Original set of Haar features [14].

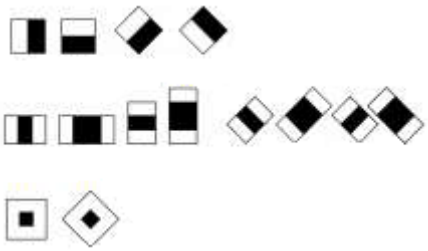

Fig. 3 Extended set of Haar features [18]

For a fast and simple calculation of Haar features, the Viola-Jones algorithm uses integral image (Equation 1), where ii(x,y) is the integral image and i(x', y') is the original image [17, 18].

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y') \tag{1}$$

The sum of the values of all pixels that lie in the up and left directions is represented by position ii(x,y) in the image. Only three mathematical operations are needed to calculate the brightness of any part of the image. The sum of the individual positions (Figure 4) in the integral image A, B, C, D is defined by the following formula:

$$S_{\Sigma} = A - B - C + D \tag{2}$$

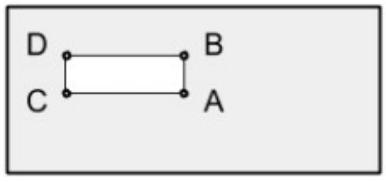

Fig. 4 Sum in integral image [18].

The most common Haar symptoms include two types that can be used separately or combined. Local Binary Patterns, that are used for texture classification in the image, and Histogram of Oriented Gradients, that uses the method of edge-oriented gradients. It is assumed that only a fraction of all Haar features can be combined into an effective classifier.

For this reason, the AdaBoost algorithm is used to extract features and then learn a cascade classifier, which consists of several weak classifiers. This set of weak classifiers is sequentially arranged and boosted, which results in a high classification capability (Figure 5) [14], [17], [18].

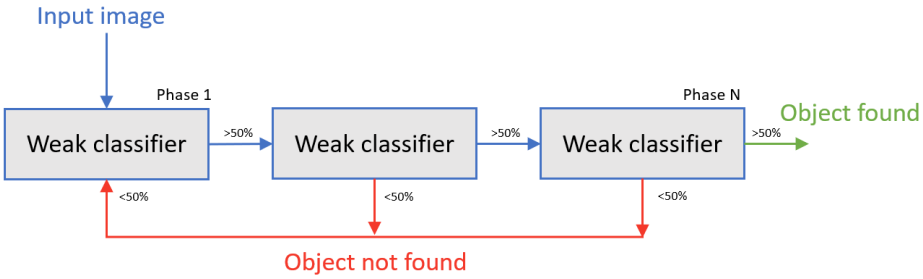

Fig. 5 Cascade classifier [17].

The input of the classifier are readings of the individual Haar features. The output is a class representing the wanted object or a class representing its absence. To use this method, a training database is required, which contains thousands of images of a given object and images that do not contain the searched object. Algorithm AdaBoost defines the weights for each classifier during the training [14, 17, 18].

Neural networks are among the most precise methods to detect human faces from the images or video; however, the system must be pre-trained. The neural network can detect relations between images and thousands of repeats of this evaluation will resolve in high quality face detection neural network. You can also use this approach to distinguish between specific faces [13, 14]. The authors in [19] proposed a neural network based system that synergistically detects eye-openness and yawning achieving a 99% precision. In [20], the authors aimed to create a unique method for evaluating a driver's attention by studying respiration data. The quality signal classification algorithm and a Nested LOSOCV algorithm were used to select a proper model. The unique method, TEDD, was verified on a private dataset achieving accuracy of 96.6% which proves not only a great potential of using neural networks, but also shows that the respiratory signal analysis might be an efficient method for fatigue detection.

To detect eyes from facial area, there are several possible methods available. **The Constrained Local Model (CLM)** method uses the face model that mostly learn from annotated shapes and local facial appearance information around detected landmarks. This method takes into consideration independent local appearance information around each landmark, which is easier to capture and more resilient to illumination and occlusion [21], [22]. The **Active Appearance Models (AAM)** methods are not using only information about shapes but consider appearance of a training data as well. To prevent the errors caused by global lightning, the appearance of a texture is first normalized. According to [23] the AAM method was simple in its nature, but caused lot of problems, for example low applicability in real-life situations and low robustness. Several enhancements have been proposed for the AAM method to increase its efficiency, such as the use of under-sampled textures, the adaptation of the regression matrix from the PCA projection of the texture to a lower dimension or optimizing the fitting process by inverse composition method [21, 22]. There are also methods based on regression directly learning mapping from the image appearance to location landmarks. Contrary to the AAM and CLM methods, it does not create any global facial model. We can classify regression methods into three main subcategories: direct regression (predicts facial landmarks in a single iteration), cascade regression (performs a

cascade prediction and usually requires initial positions of landmarks) and deep learning (derived from the direct or the cascade regressions) [21, 22, 24].

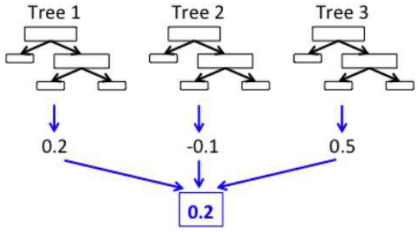

Fig. 6 Regression tree [24].

In [24], calculation-efficient method of cascade regression is introduced. Its focus was to train global regression model that can directly predict facial landmarks from image. The prediction was achieved by an ensemble of regression trees (Figure 6) which execute shape invariant feature selection while minimizing the same loss function during training as they intended to minimize at test time. A gradient boosting algorithm is implemented, too, which is susceptible to over-fitting. Over-fitting happens when a model does not generalize well from observed to unseen data [25]. Authors of [24] also stated that it is more effective to use several smaller cascade regressors than one huge cascade model.

Fig. 7 Output of the cascade regression method [24].

The individual points detected by the Cascade regression method (Figure 7) represent facial landmarks defined by the numbers 1-64 as shown in Figure 8 [24]:

- mouth 49-68
- right eyebrow 23-27
- left eyebrow 18-22
- left eye 37-42
- right eye 43-48
- nose 28-36
- facial contour 1-17

Neural Networks (NN) are increasingly used in many fields: system modelling [26], machine vision [27] or face detection field [28, 29, 30]. They consist of interconnected network of nodes – neurons. Between the individual neurons, there are weighted connections, which helps the detection of features. In [28], a hierarchical face analysis method based on deep learning is proposed. The NN structure consists of four layers: a face detector, a face part detector, a face component detector, and a face component segmentation. In [29] authors

created a three-layered cascade of deep neural network to detect facial landmarks. The method uses a set of convolution networks on each layer. The first layer provides a rough estimate of face position, the other layers refine this estimate to enhance the precision of detection. The recurrent neural network was used in [30]. The Long Short Term memory architecture was used to get set point, and the estimate was refined with a set of two parallel networks. The above-mentioned methods do have better results than the CLM, AAM or regression methods, but they also require much (often tens or hundreds of times) more training data and training time [21, 22].

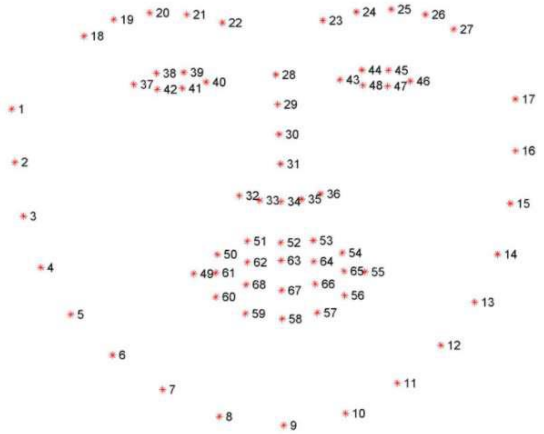

Fig. 8 Facial landmarks [24].

3 Development of a fatigue detection system

In this Section, an affordable device for driver fatigue detection is proposed. The suitable hardware and method of feature recognition is chosen, as well as software platform to develop the proposed application.

Considering the above-described image recognition methods and possible options of their implementation on a low-cost device, we decided that the easiest and most cost-effective way of implementation is to use webcam, because it does not require extensive installation compared to other methods that require one or more sensors.

As for the facial feature detection, from possible options (monitoring eyes, yawning or head tilt) eye monitoring is by far the most precise [9].

The system was algorithm was implementer in the Python programming language, and PyCharm integrated development environment. The library that provides most of the tools for image processing is OPENCV.

The solution was realized in two stages [24]:

1. Development of an algorithm for fatigue detection
2. Implementation of the algorithm on a suitable and affordable device

3.1 Development of the fatigue detection algorithm

To download the input image from the camera, the *VideoCapture* function from the OPENCV library was used. Next, using the *cvtColor* function, the image was transformed into a gray-scale. To detect the face from the image, the Viola-Jones method was applied, implemented in OPENCV library. The reasons why we decided to use the Viola-Jones method were the real-time face detection, robustness, and speed. This algorithm requires a

pre-trained face classifier at the input, which is also available in the library. Using the *CascadeClassifier()* function, the required classifier was loaded. For the actual implementation of the algorithm, the function called *face_cascade.detectMultiScale()* was used, where the processed gray-scale image is the input attribute. The output of the algorithm are the x, y, w, h values, where x represents the initial x -coordinate of the face, y represents the initial y-coordinate of the face in the image, the values w and h represent the width and height of the detected face.

To detect the eyes, we applied the approach presented in [20] which provides high-precision facial landmarks detection. Next, it was necessary to train the landmark detection model. Not all facial landmarks were to be detected, to reduce calculation time we decided to train the model only to detect eyes. The training was carried out using data from the free open iUG-300W database, which includes annotated human faces. Every image has assigned a corresponding XML file containing 68 facial landmarks points (Figure 8). To increase the speed of image processing and detection of desired facial landmarks. Using a simple script, each XML file was modified by erasing unnecessary points not related to eyes. Thus, just 12 facial landmarks are needed for each image, 6 for each eye. Then, we divided the database into two parts: a testing batch, which contained 1008 images, and a training batch, which contained 7674 images.

The XML file contained three branches:

1. <image> - path to the image
2. <box> - quadrilateral that defines the face.
    - top – represents initial coordinate Y.
    - left – represents initial coordinate X.
    - width – represents the width of the quadrilateral.
    - height – represents the height of the quadrilateral.
3. <branch> - represents the facial landmark.
    - name – represents the ID number of the landmark.
    - X – represents the X coordinate of the landmark.
    - Y – represents the Y coordinate of the landmark.

Model training was performed by the **Dlib** Python library with the function *dlib.train_shape_predictor()*with the inputs: the training database, the output file, and the additional training setup. Following options were chosen:

- *options.tree_depth = 4*: Smaller *tree_depth* values will lead to shallower trees, which are faster but potentially less accurate; larger values produce deeper, slower but potentially more accurate trees.
- *options.nu* = 0.1: floating point value ranging within <0, 1> used as a regularization parameter.
- *options.cascade_depth = 15*: this value represents the depth of the cascades; the more cascades the better the accuracy, but on the other hand a larger model is obtained.
- *options.feature_pool_size = 400*: the number of pixels used to generate the features for the random trees in each cascade.
- *options.num_test_splits = 50*: this value has an impact on the training length of the model.
- *options.be_verbose = True*: command to dump the training status.

- *options.num_threads = multiprocessing.cpu_count()*: the number of processor cores the PC has available.

In the training process some issues may be encountered. In our case, it was RAM memory overload due to the amount of processed data. We resolved it by enabling the RAM swapping if overloaded. After the training was completed, we tested the model and eye detection from the image; the achieved error rate 7,5%, was considered sufficient.

The proposed procedure evaluates data from the image and estimates the fatigue of the driver depending on the eye-opening rate. Six landmarks correlating with every eye were detected, thus the Euclidean distance between the A and B points can be calculated as follows:

$$d_{(A,B)} = \sqrt{(A_x - B_x)^2 + \left(A_y - B_y\right)^2} \tag{3}$$

where $d_{(A,B)}$ is the distance and A_x and B_x are x coordinates and A_y and B_y are *y* coordinates of the points, respectively (Figure 9).

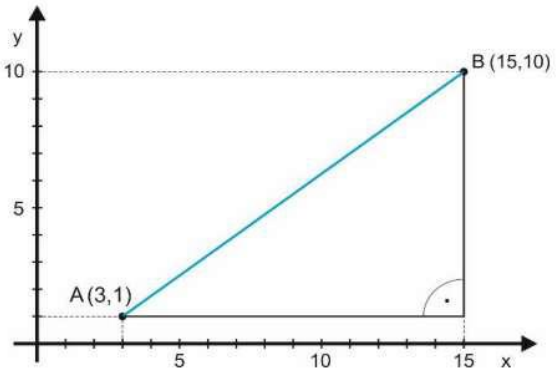

Fig. 9 Euclidean distance calculation example.

This scenario works for two points, however, to find out to which extent the eyes are opened we need to calculate the width/height ratio (eyes open rate, EOR) of the landmarks delineating the eye (Figure 10):

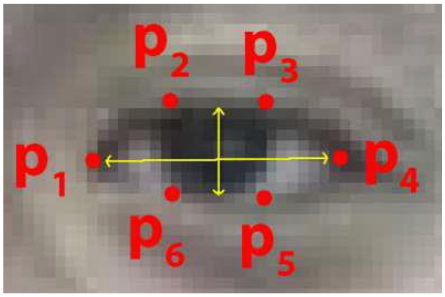

Fig. 10 Facial landmarks outlining the eye.

$$EOR = \frac{\left\| \sqrt{(p_{2x} - p_{6x})^2 + \left(p_{2y} - p_{6y}\right)^2} + (p_{3x} - p_{5x})^2 + \left(p_{3y} - p5_y\right)^2 \right\|}{2\left\| (p_{1x} - p_{4x})^2 + \left(p_{1y} - p_{4y}\right)^2 \right\|} \tag{4}$$

where *p* are respective eye landmarks (points). The logic of the system is depicted by the flow chart in Figure 11.

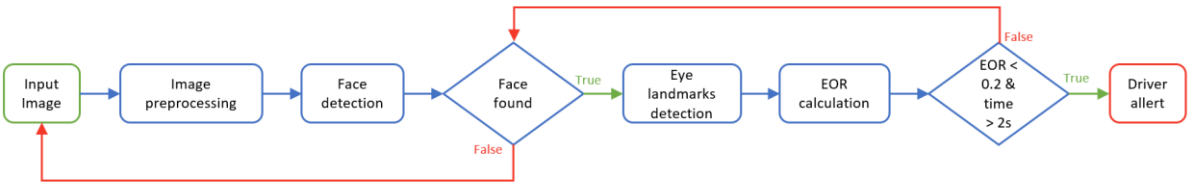

Fig. 11 Flow chart of the fatigue detection assistant system.

If we want this system to be applicable in real life, it has to operate in a real-time and be compact enough to fit into a car without obscuring the drivers' field of vision nor impacting the comfort of the ride.

Our first experiment was with **Raspberry PI 3 B+,** because it runs with Linux OS, supports Python, OPENCV, and the Dlib is compatible with Raspberry Pi. The drawback is that the board does not come with embedded video recording device. We decided to use the external webcam **Creative Live! Cam Sync HD** connected by USB. Firstly, we used SD Card to install the Raspberry Pi operating system and required libraries. Afterwards, the tested the prepared application and found out that the above chosen hardware was not sufficient to operate in a real-time and the frames per seconds were only reaching 2 FPS. There are three possibilities to deal with this issue: optimizing the code (it is questionable if the speed boos would be sufficient), selecting different algorithm (this move would require lot of additional research and time), or find more powerful and still affordable device to run proposed algorithm.

We decided to explore the latter option, because that way most of the codebase could be preserved. After examining other low-cost options, we concluded that a smartphone is quite powerful device that nearly everyone carries with and hence is available any time. To develop an application for the most widespread mobile operating system – Android, we had to resolve compatibility issues, because the application was created in Python. As the *Android Studio* with *Android Native Development Kit* (NDK) supports programming languages as Java, Kotlin and C++, the main remaining issue was the library support. The OPENCV is supported by Android applications, but the Dlib not. The substitution tool that satisfied our requirements was *Chaqupoy*.

The solution with the Android-based smartphone achieved a speed about 7 FPS. Performance of individual setups (Smartphone vs Raspberry Pi + additional camera) is compared in Table 1.

Table 1. Comparison of performance.

	Samsung Galaxy A20e A202F Dual SIM	**Raspberry Pi 3 B+**	**Creative Live! Cam Sync HD**
Camera	8 Mpx		2.1Mpx
RAM	3GB	1GB	
Processor	Samsung Exynos 7884 8-core, 1.6 GHz	Broadcom Quad-Core BCM2837B0, 4 cores, 1.4 GHz	

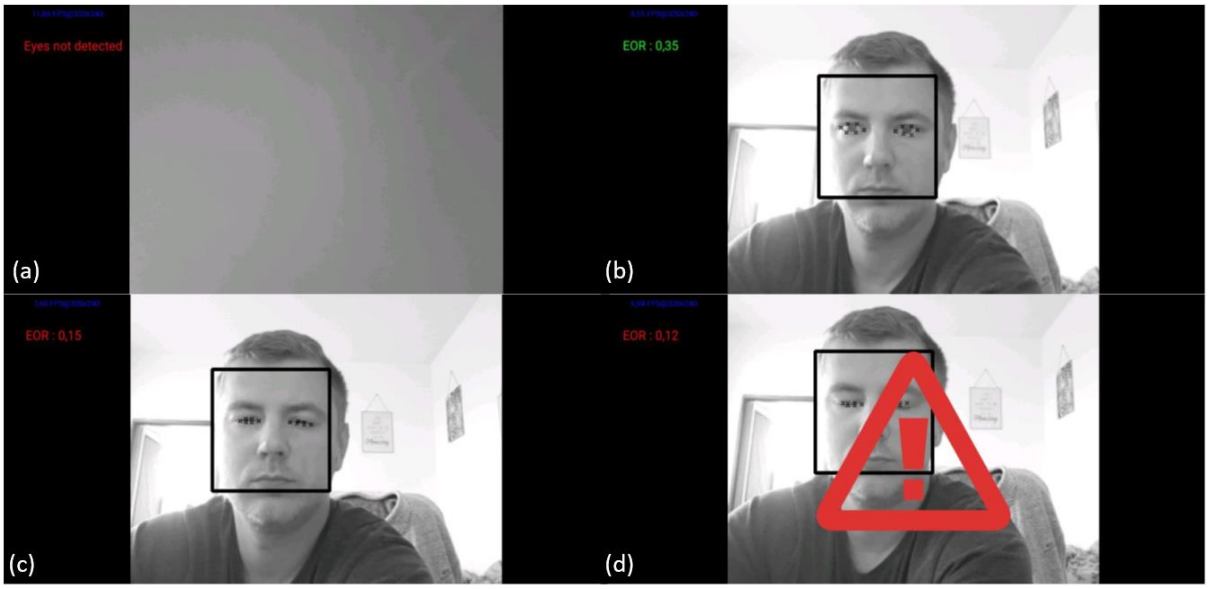

Fig. 12 Outputs of an application: (a) Eyes not detected; (b) EOR>0.2; (c) EOR<0.2; (d) Drowsiness detected.

We tested the system on the Samsung Galaxy A20e A202F Dual SIM device. The workflow was similar to previous steps (image preprocessing, face detection, eye detection, data processing).
Based on the EOR, the eye closure is classified into four states (Figure 12.):

- State 1 - Eyes not detected.
- State 2 - EOR > 0.2.
- State 3 - EOR < 0.2.
- State 4 – Drowsiness detected.

The EOR level is written also in the top left corner of the image on the screen. If the State 3 is persisting for an excessive amount of time, the system switches to State 4 sending visual and sound warning and evaluates the situation as dangerous.

DISCUSSION

The article addresses design and implementation of a simple, low-cost, on-board driver assistance system for driving fatigue detection. Ease of use, availability and affordability of the device were the main criteria which a smartphone meets the best. So, the system was developed as an Android application. The developed application employs the Viola-Jones method for feature detection and the cascade regression to monitor the EOR of the driver. Evaluating the monitored data in real time, the system can detect increase in fatigue and warn the driver of a dangerous situation about their condition. The application was so far successfully tested in laboratory conditions and testing in the real-life conditions is also planned for future.

The designed driver fatigue detection assistance system uses a trained model for eye detection from a camera image. The trained model was successfully tested on 1008 test

images, achieving a 7.5% error rate of the eye area detection. The actual testing of the assistance system was performed using an Android device. For an input image resolution of 320x240 we achieved an average of 7 processed frames per second. Testing the system on a real-time video using another Android device (One plus 6T), we achieved even a higher speed at an average of 10 processed frames per second for the same input image resolution. This proves that the developed application is usable on a wider selection of Android smartphones. Future research will focus on improving reliability by further testing the app in different cars and on different devices (smartphones).

ACKNOWLEDGEMENTS

The paper was supported by the Slovak Scientific Grant Agency grants No. VEGA 1/0637/23 and 1/0107/22. Special thanks go to Juraj Kurilla for carrying out the experiments.

REFERENCES

[1] Zhang, H., Ni, D., Ding, N., Sun, Y., Zhang, Q., Li, X. "Structural analysis of driver fatigue behavior: A systematic review", Transportation Research Interdisciplinary Perspectives 21, **2023**. DOI: 10.1016/j.trip.2023.100865

[2] El-Nabi, S. A., El-Shafai, W., El-Rabaie, E. S. M., Ramadan, K. F., Abd El-Samie, F. E., Mohsen, S. "Machine learning and deep learning techniques for driver fatigue and drowsiness detection: a review", Multimed. Tools Appl., **2023**. DOI: 10.1007/s11042-023-15054-0

[3] Ueno, H., Kaneda, M., Tsukino, M. "Development of drowsiness detection system", **2002**. DOI: 10.1109/vnis.1994.396873

[4] Saini, V. "Driver Drowsiness Detection System and Techniques : A Review", Int. J. Comput. Sci. Inf. Technol. 5 (3), **2014**.

[5] Chai, M., Li, S. W., Sun, W. C., Guo, M. Z., Huang M. Y. "Drowsiness monitoring based on steering wheel status", Transp. Res. Part D Transp. Environ. 66, **2019**. DOI: 10.1016/j.trd.2018.07.007

[6] Rongben, W., Lie, G., Bingliang, T., Lisheng, J. "Monitoring mouth movement for driver fatigue or distraction with one camera", **2004**. DOI: 10.1109/itsc.2004.1398917

[7] Ji, Q., Zhu, Z., Lan, P. "Real-time nonintrusive monitoring and prediction of driver fatigue", IEEE Trans. Veh. Technol. 53 (4), **2004**. DOI: 10.1109/TVT.2004.830974

[8] Čolić, A., Marques, O., Furht, B., "Driver Drowsiness detection System and Solutions", **2014.**

[9] Of, O., Carriers, M. "PERCLOS : A Valid Psychophysiological Measure of Alertness as Assessed by Psychomotor Vigilance", October 31 (5), **1998**.

[10] Škoda auto s.r.o. "Fatigue recognition assistant", [Online] Available at: https://eshop.skoda-auto.sk/en_SK/fatigue-recognition-assistant/p/5E0054801

[11] Volkswagen Group, "The Transporter 6.1 Optional Extra Brochure", **2020**. [Online]. Available at: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiq6sCk6IKEAxWq0AIHHWNyDBUQFnoECDQQAQ&url=https%3A%2F%2Fwww.volkswagenvans.ie%2Fidhub%2Fcontent%2Fdam%2Fonehub_nfz%2Fimporters%2Fie%2Fmodels%2Fdownloads%2Foptional-ext

[12] I. Rear View Safety, "iVue Driver Fatigue System". [Online]. Available at: https://www.rearviewsafety.com/driver-fatigue-system-rvs-335.html

[13] Kumar, A., Kaur, A., Kumar, M. "Face detection techniques: a review", Artif. Intell. Rev. 52 (2), **2019**. DOI: 10.1007/s10462-018-9650-2

[14] Hatem, H., Beiji, Z., Majeed, R. "A Survey of Feature Base Methods for Human Face Detection", Int. J. Control Autom. 8 (5), **2015**. DOI: 10.14257/ijca.2015.8.5.07

[15] Anila, S., Devarajan, N. "Simple and Fast Face Detection System Based on Edges", Int. J. Univers. Comput. Sci. 1 (2), **2010**.

[16] Yao, J., Cham, W. K. "Efficient model-based linear head motion recovery from movies", in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2, **2004**. DOI: 10.1109/cvpr.2004.1315193

[17] Jones, M., Viola, P. "Fast multi-view face detection", Mitsubishi Electr. Res. Lab TR-20003-96 3.14, **2003**.

[18] Viola P., Jones, M. "Rapid object detection using a boosted cascade of simple features", in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1, **2001**. DOI: 10.1109/cvpr.2001.990517

[19] Ahmed, M. I. B., et al., "A Deep-Learning Approach to Driver Drowsiness Detection", Safety 9 (3), **2023**. DOI: 10.3390/safety9030065

[20] Guede-Fernández, F., Fernández-Chimeno, M., Ramos-Castro, J., García-González, M. A. "Driver Drowsiness Detection Based on Respiratory Signal Analysis", IEEE Access 7, **2019**. DOI: 10.1109/ACCESS.2019.2924481

[21] Wu, Y., Ji, Q. "Facial Landmark Detection: A Literature Survey", Int. J. Comput. Vis. 127 (2), **2019**. DOI: 10.1007/s11263-018-1097-z

[22] Ferkova, Z., Matula, P. "Multimodal Point Distribution Model for Anthropological Landmark Detection", in Proceedings - International Conference on Image Processing, ICIP 2019, September, **2019**. DOI: 10.1109/ICIP.2019.8803252

[23] Wang, N., Gao, X., Tao, D., Yang, H., Li, X. "Facial feature point detection: A comprehensive survey", Neurocomputing 275, **2018**. DOI: 10.1016/j.neucom.2017.05.013

[24] Kazemi, V., Sullivan, J. "One millisecond face alignment with an ensemble of regression trees", **2014**. DOI: 10.1109/CVPR.2014.241

[25] Milesich, T., Danko, J. Bucha, J. "Neural Networks - A Way to Increase the Fuel Efficiency of Vehicles", Strojnícky časopis – Journal of Mechanical Engineering 68 (1), pp. 81 – 88, **2018**. DOI: 10.2478/scjme-2018-0008

[26] Mishra, A. "Machine Learning Algorithm for Surface Quality Analysis of Friction Stir Welded Joint", Strojnícky časopis – Journal of Mechanical Engineering 70 (2), pp. 11 – 20, **2020**. DOI: 10.2478/scjme-2020-0016

[27] Ying, X. "An Overview of Overfitting and its Solutions", in Journal of Physics: Conference Series 1168 (2), **2019**. DOI: 10.1088/1742-6596/1168/2/022022

[28] Luo, P., Wang, X., Tang, X. "Hierarchical face parsing via deep learning," **2012**. DOI: 10.1109/CVPR.2012.6247963

[29] Sun, Y., Wang, X., Tang, X. "Deep convolutional network cascade for facial point detection", **2013**. DOI: 10.1109/CVPR.2013.446

[30] Chen, Y., Yang, J., Qian, J. "Recurrent neural network for facial landmark detection", Neurocomputing 219, **2017**. DOI: 10.1016/j.neucom.2016.09.015