Yawning Detection Using Embedded Smart Cameras

Mona Omidyeganeh[1], Shervin Shirmohammadi[1], Shabnam Abtahi[1], Aasim Khurshid[2], Muhammad Farhan[2], Jacob Scharcanski[2], Behnoosh Hariri[1], Daniel Laroche[3], Luc Martel[3]
[1] Distributed and Collaborative Virtual Environments Research Laboratory, University of Ottawa, Ottawa, Canada
{m_omid | shervin | sabtahi | bhariri} @discover.uottawa.ca
[2] Instituto de Informática and Graduate Program on Electrical Engineering, UFRGS, Porto Alegre, Brazil
{akhurshid | jacobs}@inf.ufrgs.br
[3]CogniVue Corp., Gatineau, Quebec, Canada, {dlaroche | lmartel}@cognivue.com

***Abstract*— Yawning detection has a variety of important applications in driver fatigue detection, well-being assessment of humans, driving behaviour monitoring, operator attentiveness detection, and understanding the intentions of a person with a tongue disability. In all of the above applications, automatic detection of yawning is one important system component. In this paper, we design and implement such automatic system, using computer vision, which runs on a computationally-limited embedded smart camera platform to detect yawning. We use a significantly modified implementation of the Viola-Jones algorithm for face and mouth detection, and then use back projection theory for measuring both the rate and the amount of the changes in the mouth, in order to detect yawning. As proof-of-concept, we have also implemented and tested our system on top of an actual smart camera embedded platform, called APEX™ from CogniVue Corp. In our design and implementations, we took into consideration the practical aspects which many existing works ignore, such as real time requirements of the system, as well as the limited processing power, memory, and computing capabilities of the embedded platform. Comparisons with existing methods show significant improvements in the correct yawning detection rate obtained by our proposed method.**

***Index Terms*— yawning detection, vision based measurement, smart camera, embedded vision algorithm, low complexity detection.**

I. INTRODUCTION

RELIABLE and automatic yawning detection is a requirement of a number of important applications. The most common usage of yawning detection is in driver fatigue detection systems, where yawning is one factor among others such as percentage eye closure, eye blink rate, blink speed and amplitude, head motion, and the driver's direction of attention [1, 2]. The reason for so much interest in driver fatigue detection is the proven correlation between driver fatigue and a significant increase in the probability of car accident [3, 4 and 5]. Once fatigue has been detected, a variety of actions can be taken to help the driver, such as playing an audible warning sound, rendering vibrations on the steering wheel and/or driver's seat, displaying messages, or supplying more oxygen to the driver, for example by paced breathing using a pulse sound synchronized with heartbeats [6]. Driver behaviour monitoring systems, which may or may not include driver fatigue detection, also rely on yawning detection as one factor of determining the driving behavior [7].

Yawning detection is also used for in-home health care systems, such as intelligent mirrors, which take into account yawning as one of the factors to determine a person's health status, to improve the person's life-style via tailored user guidance [8]. Operator attentiveness is another application that uses yawning detection as one of a few deciding factors in determining whether or not an operator of a critical system such as heavy machines, nuclear reactor controls and monitors, air traffic controllers, etc., is paying attention to the operation or not [9]. Finally, yawning detection can also be used in systems that determine the communication intentions of a person with a tongue disability, specifically to detect false estimation [10].

For all of the above systems, which require automatic detection of yawning, the cost of the system is very important in order to make it economically viable. Vision Based Measurement (VBM) can help [11]. In VBM, a camera or optical sensor is used to acquire an image of a physical scene, and the image is then processed in an operations unit to detect and/or measure a specific subject of interest. The camera and the operation unit are together known as a smart camera. Since such systems are becoming more and more affordable every day, VBM systems are now being considered a practical solution for applications such as detection of human physical features such as face and iris [12, 13, 14 and 15] or automotive assistive systems [16].

In this paper, our goal is to develop a real time system using a smart camera that detects a yawning mouth with high detection rate. Due to our research collaboration with CogniVue Corp., who manufacture the APEX™ embedded smart camera, an absolute requirement of our system was that it must be able to work on the embedded hardware with computationally-limited capabilities. As we shall see in Section III, this limitation had a significant effect in our methodology, and led us to specific design choices which

differ from other yawning detection systems. In addition, the collaboration with CogniVue led to our proof-of-concept being tested in a car setting, although it must be noted that our algorithms for yawning detection are not restricted to a car. Our system's design involves several steps, including the real time detection of the person's face, detection of the mouth, and detection of yawning. Figure 1 shows the overall algorithm of our system. In [17], we presented a snapshot of our design and implementation with the camera on the dashboard in front of a driver. In this paper, we explain our design and implementation with more details, and we also extend our design to accommodate situations where the camera must be installed under the front mirror. More experiments and quantitative comparisons with other existing approaches are also reported.

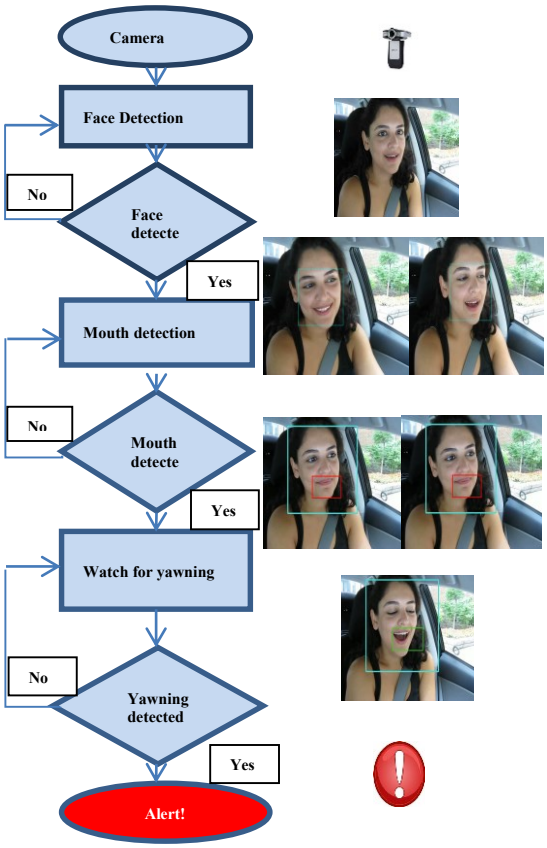

Fig. 1. Yawning Detection Algorithm.

The rest of this paper is organized as follows. After discussing the related work in Section II, Section III describes our proposed approach, while Section IV analyzes the computational complexity of our approach. Section 0 presents all implementations, including Viola Jones, our proposed method, and other system part implementation, and also describes the hardware specifications and limitations that we were faced with in our design. Experimental results and comparisons with other representative methods are in Section VI, and finally conclusions are made in Section VII.

II. Literature Review

To detect yawning, the mouth itself must be detected first. This can be done by detecting the face, and then detecting the mouth. There are many existing approaches to detect face and/or mouth. In [18], the authors propose an embedded system which detects face location by analyzing the movement of eyes. The eyes are detected in sequential input images by determining the variation of their region. Another embedded approach is introduced in [19], which utilizes a Genetic Algorithm with AdaBoost training to optimize the detection performance. A different approach for open or closed mouth detection using static supervised classification based on log-polar signatures is presented in [20]. The transformation-based nature of the approach tends to lose the spatial information, so grading mouth opening is challenging even after the optimization of the SVM classifier, as we will show in Section V. In [21], the authors propose a real-time face detection algorithm for locating faces in images and videos. This algorithm finds not only the face regions, but also the precise locations of the facial components such as eyes and lips. Different color based methods for mouth and lips detection have also been proposed in [22, 23], and most of them suggest that color is not a good feature for face and lips or mouth detection when substantial changes in illumination and head pose are expected. In [24] an embedded system for face detection which focuses on the use of FGPAs is proposed. The work in [25] takes advantage of grey projection and Gabor wavelets to detect the mouth corners and uses Latent Dirichlet allocation (LDA) to find a linear combination of those features to detect the yawning mouth, while [26] detects the face using the Viola-Jones technique and extracts the mouth region, in which lips are searched for through spatial fuzzy c-means (s-FCM) clustering. The proposed system in [27] requires the use of two cameras: a low resolution camera for the face and a high resolution one for the mouth. It uses Haar-like features to detect the driver's mouth, and yawning is detected based on the aspect ratio of the mouth. The work in [28] monitors driver drowsiness based on a combination of eyes and mouth gestures, and determines the state of the mouth and eyes by analyzing their feature points using Back Propagation neural networks in order to check for the conditions that involve driver drowsiness. The system in [29] uses the cascade of classifier as proposed by the Viola-Jones face detection technique, and then uses an SVM to train the classifier with the mouth features in yawning condition. Another approach to facial movement analysis has been proposed in [30] by using Adaboost and multinomial ridge regression to train the classifier of different facial actions such as blinking and yawn motions.

Despite some good results, most of the above techniques have a high computational complexity and cannot satisfy the real-time requirements of resource-limited embedded smart camera platforms. They are either lab reports without an actual field deployment, or use a high-end laptop/desktop to run their method, which is far from a practical and economical smart camera. So despite the considerable research mentioned above, today only a few yawning monitoring systems exist in some luxury cars that still suffer from a high rate of false positive detection and do not have sufficient accuracy [31].

There are three main reasons why existing techniques are not robust enough for a production-grade and commercial consumer system: computational complexity, facial obstruction, and lighting conditions [32]. In this paper, we address the first challenge by proposing a design and implementation that does consider practical limitations and can work in real smart cameras.

Our architecture and method started with the work in [32], which worked well because it was detecting and tracking multiple features including, face, eyes, and mouth, and had good robustness. But, similar to existing systems, it turned out to be too computationally complex to be implemented in an actual embedded smart camera platform. In this paper, we only do face and mouth detection, with no tracking, and we do not use complex algorithms, in order to have a reasonable computational footprint. We also avoid using techniques that require a large training database of yawning based on classifiers, in order to decrease the computational time. Finally, we have tried to maintain a high level of detection efficiency when optimizing other aspects of the system such as complexity and ease of implementation. All of these have caused our design to take an approach significantly different from the related work. With this in mind, we now describe our proposed approach in detail.

III. Proposed Approach

The overall approach was shown in Figure 1, where the first step is to detect the person's face. Face detection can be challenging because faces are non-rigid and have a high degree of variability in size, shape, color and texture. In our case, we assume that the camera is facing the person at a fixed angle, as shown in Figure 10. Therefore, the problem of relative camera-face pose is less challenging in our case while head position might still vary from case to case. There is also a great deal of variability among faces including shape, color and size. One of the most functional face detection methods is the Viola-Jones algorithm [33] which has been already implemented in the OpenCV software library. We use this algorithm as a guideline for our own face and mouth detection methods. Therefore, before explaining the details of our system, we first briefly discuss the Viola-Jones face detection algorithm, and explain our design and how it differs from the typical Viola-Jones method.

A. Viola-Jones Face Detector

The method of object detection using the Viola-Jones theory is capable of processing images very rapidly while achieving a high detection rate. There are three main techniques involved with this detector. The first technique is an integral image, which is useful for fast feature evaluations and decreasing the complexity of feature detection for each frame. The second technique is a process for creating a classifier by selecting a small number of features using Adaboost. The last technique is a method of combining classifiers in a cascade structure.

The first step of the Viola-Jones algorithm uses diagonal features as well as other types of Haar-like features to extract face features. Figure 2 shows the types of different Haar-like features used by the classifier for face detection. The algorithm of integral image speeds up the computation of the following Haar-like features.

The value of the integral image at any point (x, y) contains the sum of all the pixels above and to the left of (x, y) inclusive:

$$\mathbf{ii}(\mathbf{x},\mathbf{y}) = \sum_{\mathbf{x}' <x,\mathbf{y}' <y} \mathbf{i}(\mathbf{x}' ,\mathbf{y}') \quad (1)$$

Where $ii(x, y)$ is the integral image and $i(x', y')$ is the original image.

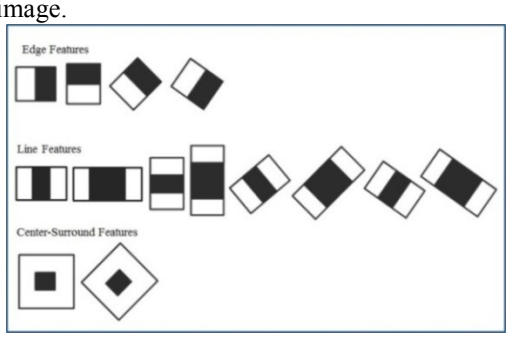

Fig. 2. Different Haar-like features, reproduced from [33].

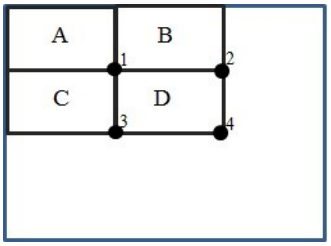

Fig. 3. Haar-like features, reproduced from [33].

Furthermore, the summed area can be calculated in a single pass over the image by considering the fact that the value in the summed area at (x, y) is:

$$\mathbf{ii}(\mathbf{x},\mathbf{y}) = \mathbf{i}(\mathbf{x},\mathbf{y}) + \mathbf{ii}(\mathbf{x}-\mathbf{1},\mathbf{y})\mathbf{ii}(\mathbf{x},\mathbf{y}-\mathbf{1}) - \mathbf{ii}(\mathbf{x}-\mathbf{1},\mathbf{y}-\mathbf{1}) \quad (2)$$

After computing the summed area, any one of the Haar-like features can be evaluated in constant time at any scale or location with just four array references, as shown in Figure 3:

$$\sum_{\substack{\mathrm{A(x)}<\mathrm{x}'<C(\mathrm{x}) \\ \mathrm{A(y)}<\mathrm{y}'<c(y)}} \mathrm{i}(\mathrm{x}',\mathrm{y}') = \mathrm{ii(A)} + \mathrm{ii(C)} - \mathrm{I(B)} - \mathrm{I(D)} \quad (3)$$

In the second step, for training the T weak classifiers, a boosting algorithm is required. The AdaBoost algorithm is utilized in the system to select critical features that play the most important role in the classification decision and train the classifier. Adaboost learning algorithm is required to combine a collection of weak classification functions to form a strong classifier. It is composed of decision trees with at most a few (three) levels of splits in most cases. In the final decision making procedure, a weighted vote is assigned by each of the classifiers.

Each split is determined by whether the value v of the particular feature f is below or above the threshold t; i.e.:

$$f = \begin{cases} +1 & V \geq t \\ -1 & V < t \end{cases} \quad (4)$$

The threshold value is set in the first pass through training the data set, which classifies the input in the best way. The resulting error is used by boosting to determine the weight vote. Then, the feature is re-weighted low or high based on the correctness of classification.

The main goal of the last part is to ensure that simpler classifiers are constructed to reject the majority of sub-windows before using more complex classifiers in order to achieve low false positive rates. Proposed Face Detection Method

Viola-Jones method, if implemented as above, is not efficient enough to run in real time on smart cameras. To overcome this limitation, we introduce here a design for a fast and memory efficient face detection algorithm based on Viola-Jones. In the first step of our face detection method, all the provided data of the trained features in each node is extracted and stored in five separate files in the smart camera in order to save computation time. These files, which are the results of the training features, are used later in the detection algorithm. This way, the monitoring system can utilize the saved values instead of training the classifier and applying the integral image to find the features from the beginning. Each file contains one of the following groups of data: feature coordinate, feature threshold, feature value, stage classifier and feature weight.

For face detection, the algorithm must be able to detect faces with different sizes, not only because various people have different face sizes, but also because a person might be closer to or further away from the camera at given times, leading to larger or smaller face sizes, respectively. To do so, the features must be scaled. The details about this scaling procedure will be shown in Section IV heading C. One of the important advantages of our method is that, unlike the typical OpenCV implementation of Viola Jones, our system stops searching for another face after finding the first face in each frame as we are interested particularly in the face of the main subject and other faces are not considered so the face search time is optimized. The assumption made is realistic since the position of the subject is always the closest to the camera, so that face will be detected first by our proposed system. The faces of other people will be ignored, as they are not as big as the subject's face. As such, our system can find the subject's face when there are multiple people in the scene. Figure 4 shows some examples of this in a car driving case, where the driver's face, and in the next step his mouth are detected perfectly. In addition to saving the trained features in the files to be used directly by the camera and avoiding the training of the face detection, the result of this design also increases the speed of the face detection stage and improves the efficiency of the monitoring system. Details of the platform used in our tests and code optimizations are given under Section IV.

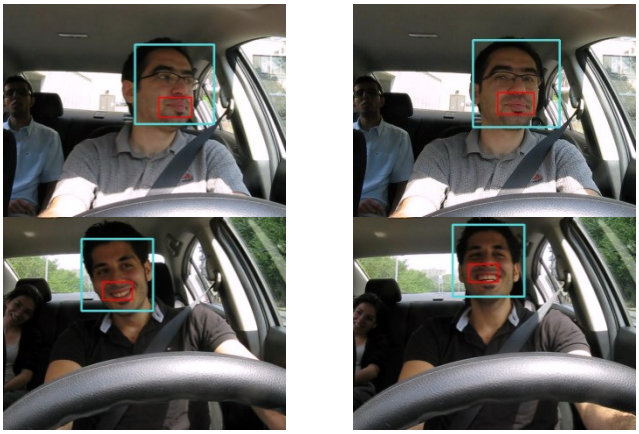

Fig. 4. Subject face and mouth detected from among multiple people.

B. *Proposed Mouth Detection Method*

After detection of the face, the mouth location must be extracted. Due to the known relative position of the person's face, the result of our mouth detection algorithm is expected to have high accuracy. Therefore, for the mouth detection algorithm, the lower half of the face is chosen as the target search region.

A similar procedure as face detection is utilized for this part. Trained features from the XML file are extracted and saved in the camera to reduce the computational time and make the system practically feasible. Searching for the mouth location starts from the upper left corner of the face frame and continues towards lower right corner. This search procedure is followed for different scales of the mouth. Instead of taking the average of 20 mouth candidates, as per the procedure of OpenCV, we use the biggest candidate to have a higher chance of finding the mouth in the frame. After locating the biggest mouth in the frame, the data related to the mouth location and size is passed to the yawning detection step, described next.

C. *Proposed Yawning Detection Method*

The last part of the process is to determine yawning. The first step here is to calibrate the system. This function is done by finding the location and histogram of the driver's mouth in the first frame. To do so, the color image for each mouth detected in the video is converted to gray scale image, as shown in Figure 5.

Fig. 5. Yawning Sequence.

The histogram of the gray scale image is obtained by counting the number of times each gray level occurs in the

image array. The histogram of the normal closed mouth position in the first frame will be saved as a reference for further calculations. To determine yawning, back projection theory is used. The basic idea in back projection theory is to create a similar image giving the similarity of each pixel of the candidate object to be matched (the candidate) with the object of interest (the reference). Generally, the features used for back projection are intensity values of the gray scale image. In order to calculate the back projection, the histogram of the reference image (in this case the normal closed mouth in the first frame) is computed and compared with the calculated histogram of candidate mouth region in the following frames. In this case, the measurement results from an image at each location over the specific region of interest are taken to form a multi-dimensional normalized histogram array by sampling from the image array. In our approach, in each video frame, we select the location of the mouth and then convert it into an image array over a chosen region of interest. Then, the histogram bin is determined for each of the arrays that are related to the mouth region. The calculated new mouth histogram is compared to the reference mouth histogram. This process is repeated for the mouth region of the entire video sequence in real-time. An appropriate threshold is selected experimentally and used to convert the gray scale image to black and white based on the back projection concept. After the conversion of the input gray scale frame to a black and white (binary) image, the system checks if there is yawning in that frame. Yawning is detected by comparing black and white pixels, and two basic conditions must be satisfied: (a) the ratio of the number of black pixels in the current and in the reference frame must be greater than a threshold value (as in Eq 5), and, (b) the ratio of the black pixels in the mouth region and white pixels in the region around the mouth must be greater than a second threshold (as in Eq 6). If both conditions are satisfied simultaneously, then the system detects this particular frame as showing yawning, and this process is repeated for the subsequent frames of the video. The equations are:

$$\frac{NBC}{NBR} > Th1 \quad (5)$$

$$\frac{NBC}{NWC} > Th2 \quad (6)$$

where NBC is the total number of the black pixels in the mouth block of the current frame, $NBR$is the total number of the black pixels in the mouth block of the reference frame, and $NWC$is the total number of the white pixels in the mouth block of the current frame. $Th1$ and $Th2$ are thresholds defined experimentally. Their assessment using Receiver Operating Characteristic (ROC) curves is described in Section IV heading E. It should be noted that for yawning detection, we are employing the image of the mouth and in this stage the face has already been detected and the luminosity changes of the image do not affect this part of the process. Also, the grayscale histogram has been used since the mouth opening is estimated based on the ratio between black and white pixels, and one of our goals is to reduce the computational complexity and have a fast algorithm that is able to perform in real-time.

IV. Computational Complexity Analysis

To have an idea about the computational complexity of the system, in this section we describe and calculate the number of steps or operations taken by the algorithms in various parts of the system. Here, we give the computational complexity analysis of our method, and compare it to the complexity analysis of methods [20] and [21], which are also compared against our method experimentally in section VI. The main reasons for providing these computational complexity analyses are: a) the methods have been implemented and tested on different platforms with distinct processing capabilities, so comparing computing times could be misleading; and b) computational complexities are independent of the computational platform, and can provide unbiased estimates of the complexities of the methods.

For our proposed system, computational complexity is divided in two stages: training complexity and testing complexity. Training is done offline just once, so its complexity will not affect the performance of the system in action. Its stages consist of face training and mouth training, as follows:

A. Computational Complexity of the Training Stage

Computational complexity of the face detection training stage:
Weak classifiers training computational complexity (face detection) = $O(WNlog_2N)$
Parameters update computational complexity (face detection) = $O(N)$
Committee testing computational complexity (face detection) = $O(V_Plog_2V_P + V_n)$.

N=Number of face samples of the proposed method face detection training stage,
V_P =Number of positive validation examples used in the face detection training stage,
V_n =Number of negative validation examples used in the face detection training stage,
W=Number of weak classifiers used in the face detection training stage,
and $O(N)$ is required to update the weights of the training set.

Computational complexity of mouth detection training stage:
Weak classifiers training computational complexity (mouth detection) = $O(W'N'log_2N')$
Parameters update computational complexity (mouth detection) = $O(N')$
Committee testing computational complexity (mouth detection) = $O(V'_Plog_2V'_P + V'_n)$.

N'= Number of mouth samples of the proposed method mouth detection training stage,
V'_P = Number of positive validation examples used in the mouth detection training stage,

V_n' = Number of negative validation examples used in the mouth detection training stage,
W'= Number of weak classifiers used in the mouth detection training stage,
and $O(N')$ is required to update weights of the training set.

Overall computational complexity of the training stage:

$$O(WNlog_2N + N + V_Plog_2V_P + V_n + W'N'log_2N' + V'_Plog_2V'_P + N' + V_n') = \boldsymbol{O(WNlog_2N + V_Plog_2V_P + W'N'log_2N' + V'_Plog_2V'_P)}$$

B. Computational Complexity of the Testing Stage

In the testing stage, each time a test image is received, the face and the mouth are detected and then yawning/non-yawning are verified. So the computational complexity is given by summation of the computational complexities of all these three stages. Next, we analyze the computational complexities of these stages, one by one.

Computational complexity of the face detection testing stage:
Computational complexity = $O(IMS)$
where:
I=size of the input image
M= number of selected face features
S=size of the input face block
For a test image, the first face block is selected to find if it has a face or not. All features/weak classifiers are tested with their corresponding weights. Each weak classifier requires two iterations over the input image. The computational complexity of testing *N* weak classifiers is given by $O(2NS)$. Then, the face block is shifted to the right by one pixel, and the process is repeated, and this process is applied to the entire input image with size *I*, which gives a computational complexity of $O(IMS)$.

Computational complexity of the mouth detection testing stage:
Computational complexity = $O(SM'S')$
where:
S=size of face block (i.e., the search for the mouth is done within the face block, not in over the whole image),
M'= number of selected mouth features,
S'=number of pixels in the mouth block.

Computational complexity of the yawning detection testing stage:
Computational complexity = $O(S')$,
where S' is the size of the detected mouth block. The count of black and white pixels in one frame requires S' operations (i.e. proportional to the number of pixels in the mouth block), and this operation is performed for two frames (i.e., the current and the reference frames).

Overall complexity of the testing stage:

$$\boldsymbol{O(IMS + SM'S' + S') = O(IMS + SM'S')}$$

Table I compares the complexity of our method with [20] and [21], the notation and analysis of which is shown in the Appendix.

TABLE I
COMPUTATIONAL COMPLEXITY ANALYSIS

METHOD	TRAINING	TESTING
OUR METHOD	$O(WNlog_2N + V_Plog_2V_P + W'N'log_2N' + V'_Plog_2V'_P)$	$O(IMS + SM'S')$
[20]	$O(S'K_SN_S + N_S(P^2N + P^3) + \max(N_S, d)\min(N_S, d)^2)$	$O(S'K_S + P^2S' + P^3 + S_V)$
[21]	N/A	$O(IL + GC + S')$

V. IMPLEMENTATION

This section describes the implementation of both the OpenCV method and our proposed method, as well as the specification of the platform used for our implementation.

A. Platform Specifications

The automotive APEX platform used in our work contains the CogniVue CV2203 highly-programmable Image Cognition Processors (ICP). It consists of an ARM 926EJTM 350MHz master processor, 34B Ops/sec low-power DSP subsystem using patented massively parallel Array Processor Unit (APU), a second 350MHz ARM 926 processor, H/W acceleration blocks, wide-bandwidth stream DMAs, internal dual 64-bit AXI data buses to/from all blocks, 16Mbyte DDR SDRAM, and 1Gbit NAND Flash. While it supports 96 parallel Computational Units (CUs) which allow heavy massive processing, and can encode/decode D1 MPEG4 video at 30fps, the platform does not support floating point operations, divisions, or numbers larger than 16 digits. These limitations led to specific design choices as described in Section IV and implementation choices as described next.

To overcome the lack of floating point operation, we used the fixed point operation by scaling all the floating point numbers by 2N to discard the digits after the floating point. In this case N depends on the number of the required floating point digits to maintain enough accuracy. The number 2 is selected (instead of the more intuitive 10) for computational efficiency. Scaling by powers of two has the advantage of easily shifting the bit pattern of the number to the right by 1 bit for dividing the number by 2 and shifting a number to the left by 1 bit for multiplying the number by 2. We were also careful to scale both sides of all operations, so that scaling won't affect the final results. To prevent overflow in intermediate results, due to the limited number of digits per number, we did another layer of scaling. For example, to multiply a few big numbers and then divide them by a big number, even though the final result is less than 16 bits, there is overflow in the intermediate multiplication result. Our approach was to scale those numbers down before multiplying them, and then scale up the result.

B. Implementation of Viola Jones in OpenCV

The OpenCV Viola Jones face detection function has been trained with a huge number of face and non-face images with a fixed size of 20×20 pixels. The Haar-like features were applied on these images and all the information about those features that have been selected using Adaboost from the set of all features are saved in an XML file in OpenCV. In order to detect a face, the training algorithm in OpenCV uses 22 stages of classifiers with different numbers of trees and nodes. Each stage includes a number of decision trees which gets larger resulting in stronger classifiers as we go to higher stages. For example, the first stage has 3 trees but the last stage (stage 22), has 212 trees. All decision trees have only a single node and each node provides information of one feature.

C. Implementation of our Proposed Algorithm

As was already mentioned, the Haar-like features are trained with a large number of 20×20 images and these features are saved in the XML file. The face detection algorithm utilizes these features to find the location of the face but does not necessarily detect only faces with the same size. In order to overcome this, the features must be scaled. In the case of having faces larger than 20×20, the feature coordinates and the feature weight of each node are rescaled. Therefore, the first task is to search for faces with different sizes depending on the difference between the subject's face and also their distance to the camera in the video frame. Hence, different scale ratios of the features are used to search for the face location in the implemented code. Since the scale ratio varies based on the size of the face, we designed a scale ratio table, as described next.

To design the table of scale ratio for different face sizes, various possible face sizes for different people based on the frame size were considered. For example, in the typical case of 640×480 frame size, and considering the distance of the subject to the camera in various vehicles, we assumed that the largest possible face size can be 460×460 and the smallest face size can be 100×100. Therefore, by taking this information into consideration, 18 scale ratios were assigned to detect the subject's face. In order to increase the detection speed and shorten the computation time required to scale the Haar-like features for different face sizes, all of the data related to each node in the 22 cascades of classifiers are saved in the above-mentioned files in 18 scales. In total, 2153 features are defined in the OpenCV face detection algorithm. Moreover, it should be noted that our algorithm starts scanning for the face from the corner of the image with the specific feature size in the beginning. If the classifiers reject the first location, the algorithm will shift the search location to the right with a specific step size. The same procedure will be applied for the whole image starting with the larger possible scale ratio and scanning toward smaller scales. If the system does not detect the face with that particular feature size, the next scale for the feature will be used and the starting point from the corner of the image will be assigned until the face is detected. Figures 6 and 7 show the search steps for the face in two specifically larger and smaller scales, respectively.

The step size in our system is based on the face size: it uses bigger steps for bigger face sizes in order to expedite the search process. After finding the face location in the video frame, the x and y coordinates of the left corner of the face and the height and width of the face will be saved to limit the search area for detecting the mouth location.

In summary, we have focused on implementing an optimized version of the Viola-Jones algorithm in our platform and we took advantage of two different types of

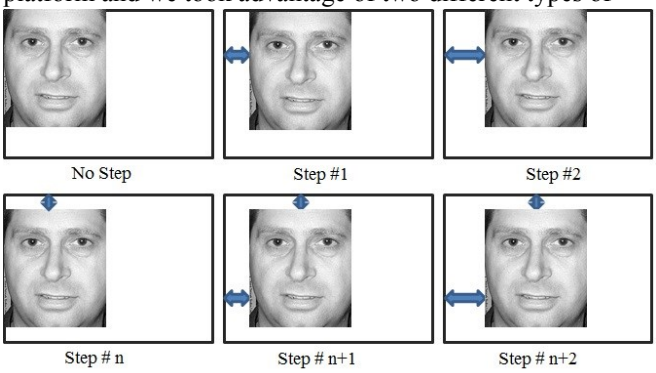

Fig. 6. Search for the face in a larger scale.

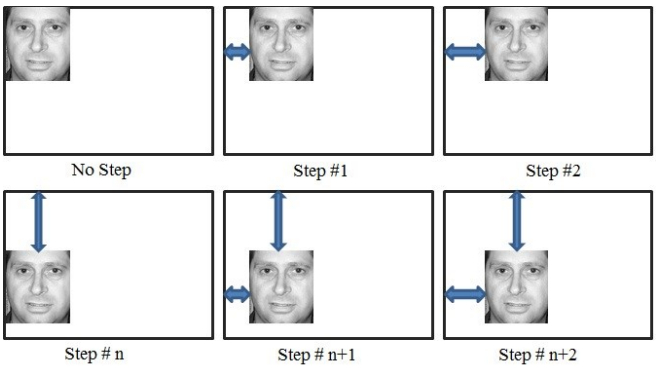

Fig. 7. Search for the face in a smaller scale.

optimizations. First, we took the C++ implementation of Viola-Jones from OpenCV open source library and analyzed the computational complexity of its building blocks. We then either re-implemented or modified some of the most computationally intensive blocks to make it more efficient. In the second optimization step, we took advantage of the hardware characteristics of our embedded system to get a better performance. As mentioned earlier, the APEX platform has an Array Processor Unit in addition to its ARM processor. The ARM processor can offload certain types of the operation to the APU in order to be able to finish things faster. We have taken advantage of the APU to be able to process frames with a faster speed.

D. OpenCV Mouth Detector implementation

This mouth detector is computed with 7000 positive samples. A large number of normal mouth images in the size of 25×15 are used to train the OpenCV classifier, and its detector contains 16 cascade classifiers each with a different number of trees. The first cascade in this case has 12 trees and the last cascade has 217 trees. The total number of features for

determining the mouth location is 1515.

E. Thresholds Assessment for Yawning Determination

The thresholds Th1and Th2 were assessed using Receiver Operating Characteristics Curves (ROC) of yawning detection in the tested videos. To determine the thresholds, the ROC plots are sketched for different thresholds showing the recall rates versus the precision values, as shown in Figure 11. Thus Th1 and Th2 values are chosen equal to 4 and 1.5, respectively, where they result the best recall and precision rates.

VI. EXPERIMENTAL RESULTS AND EVALUATIONS

In this section, we evaluate the performance of the proposed approach for face, mouth and yawning detection. We first describe our dataset, and then we explain our experiment details.

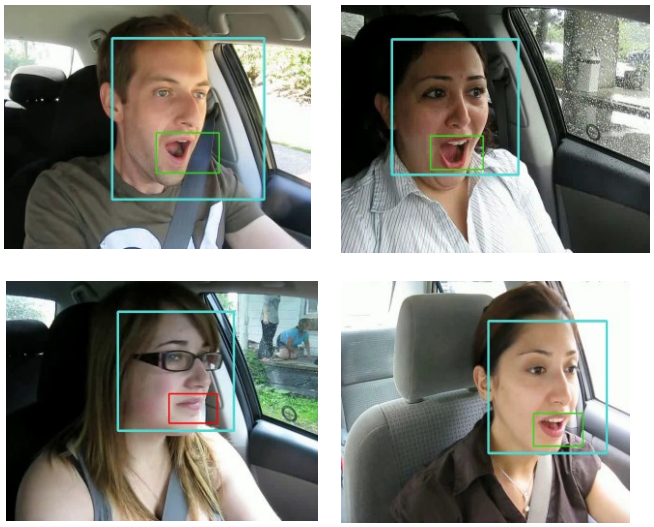

Fig. 8. Different lighting conditions and performance of our approach.

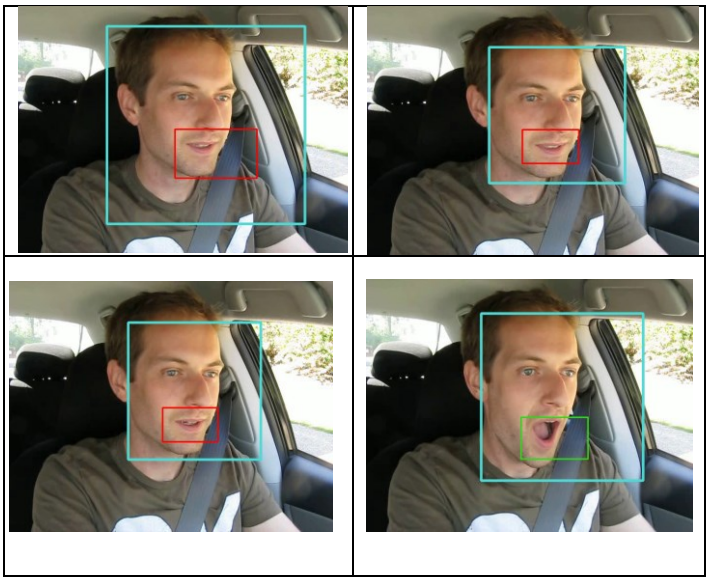

Fig. 9. Different levels of mouth openings (silence/talking), and of the performance of our approach.

A. Dataset

Our yawning dataset, called YawDD, is publically available and described in detail in [34]. This dataset was obtained from a near-realistic driving scenario, although it has also been used in other yawning detection applications, such as smart mirrors for health monitoring [8]. It consists of two sets: in one set a camera was installed under the front mirror (Case I) as shown in Figure 10a and in the other it was installed on the dash (Case II) as shown in Figure 10b, both representing typical installations in real driver monitoring systems such as APEX. Videos of people driving were collected at a resolution of 640x480 pixels, 24-bit true color (RGB), at 30 frames per second. All the videos were taken from different ages, ethnicities, and facial characteristics. Each participant performed three tasks in the video: sit and drive normally, talk or sing while driving, and yawn while driving. The second task was performed to distinguish between talking/singing and yawning, where both scenarios might lead to an open mouth and therefore a false positive might be detected. Each video was 15-40 seconds. Figure 8 illustrates some examples where the detection occurs under varying illumination conditions, showing our system's good performance. Also, different open mouth scenarios occurring in talking, singing or yawning situations are illustrated in Figure 9, and our system performance is shown in the form of a red rectangle around non yawning mouth and a green rectangle around a yawning mouth.

B. Experiment Parameters and Details

For the specific testing context in our experiments, we chose a driving scenario, although our design and implementation is generic and can be applied to any yawning detection system.

Our measurements are defined as follows: a True Positive (TP) occurs when a real yawning situation is correctly detected as yawning by the system; a True Negative (TN) occurs when a non-yawning situation is correctly detected as non-yawning by the system; a False Positive (FP) occurs when a non-yawning situation is incorrectly detected as real yawning; a False Negative (FN) occurs when a real yawning situation is incorrectly detected as non-yawning. Thus, the rate of correct yawning detection (RCD) in the video sequence is used to evaluate the system performance and is defined as follows [35]:

$$RCD = \frac{R\,(frames)}{T\,(frames)} \tag{7}$$

where *RCD* often is multiplied by 100% and expressed as a percentage (%), *R(frames)* denotes the number of frames detected correctly as yawning or non-yawning frames, and *T(frames)* is the total number of frames in the test set. The above definition is agreeable with the following expression:

$$RCD = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{8}$$

(a) Camera installed under the front mirror

(b) Camera installed on the dash

Fig. 10. Yawning Detection Algorithm.

Three methods were selected against which the effectiveness of our approach was evaluated. These are (a) the original Viola-Jones approach for mouth opening/closing detection, (b) one method representative of the transform-based approach for mouth opening detection [20], and (c) one method representative of color-based approach for face and lips detection [21].

TABLE II
DETECTION RESULTS

	Face detection CASE I	Mouth detection CASE I	Yawning detection CASE I	Face detection CASE II	Mouth detection CASE II	Yawning detection (RCD) CASE II
OpenCV	61%	48%	18%	85%	57%	20%
[20]	89%	68%	54%	94%	79%	67%
[21]	69%	52%	13%	78%	59%	19%
Proposed method	89%	68%	65%	94%	79%	75%

We tested the videos in YawDD using our proposed method running on APEX and the above approaches. The color-based lip detection approach in [21] was used to detect faces and mouths, and a yawning occurrence was then detected using the proposed method. Results are shown in Table II. The results in Table II indicate that our method can outperform existing approaches, including [20] and [21]. It shall be observed that [21] provides comparable performances for face and mouth detections in CASE II because better lip detections are obtained for female drivers since they usually wore makeups, but poorer performances were observed when the mouths were opened. The proposed method stands high in relation to the comparative approaches in all tested videos in CASES I and II. Our method works on video, detects faces, mouths and yawning as a system while the method in [20] can only be used to detect yawning when the mouth area is given as input.

The results in Table II also indicate that the face, mouth and yawning detection accuracies in CASE I are potentially lower than in CASE II, which suggests that it is more useful to install the camera on the dash instead of the front mirror. The 75% detection rate for yawning is quite satisfactory considering existing and practical systems that can actually run on smart cameras; however, 25% of the missed yawning can potentially be dangerous, and indicate that still there is need of more research. It can also be seen that the yawning

detection is affected by the face and the mouth detection, which is a research area that needs more work considering the current state of the art.

TABLE III
YAWNING DETECTION STATISTICS USING OUR METHOD

	# Frames	#Real Yawning Frames	%True Positives	%True Negatives	%False Negatives	%False Positives
Average	560	100	70%	81%	33%	7%
Maximum	2443	376	80%	94%	78%	22%
Minimum	76	41	45%	72%	19%	3%

We also ran our method on an IBM compatible PC with an Intel processor i5 3.2 GHZ (4 CPUs) and 8GB RAM using OpenCV code, where its speed was measured at 30 fps, in comparison with 3 fps on APEX. Table III reports the average, maximum and minimum number of frames of the real yawning situations in the tested videos, as well as the percentages of true positives, true negatives, false negatives and false positives, using our method. It shall be observed that the values in Table III are the average, minimum and maximum of each column, and the columns are independent. For example, if the video with the minimum number of frames is 76, the number of frames showing real yawning is not necessarily 41 in that video, because the minimum number of yawning frames may refer to a different video.

The results of recall and precision of 30000 tested frames were also calculated for different thresholds. The graph of the results for the proposed method and OpenCV method is shown in Figure 11. The face and mouth detection algorithms of OpenCV are employed along with the template matching algorithm of OpenCV to detect yawning, and the results are compared with our method.

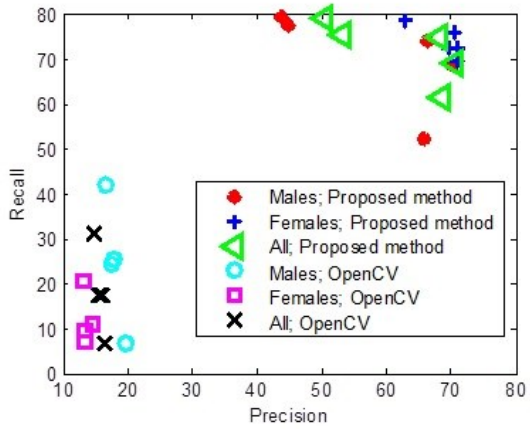

Fig. 11. Recall versus Precision for Yawning Detection.

It is interesting to note that our method outperforms OpenCV in terms of detection accuracy. One reason is that, OpenCV searches for the face/mouth location in the whole frame from the biggest possible face/mouth size to the smallest in order to find all the candidates in the image. This is not necessary in a driving scenario, as can be seen in Figure 12, where the OpenCV method has in some cases incorrectly detected objects that it thinks are faces, in addition to the driver's face. On the other hand, our system stops searching for another face/mouth after finding the first one. This functionality increases both the speed and the accuracy of the system significantly. The other reason is that, for mouth detection, OpenCV's algorithm finds around 20 candidates for the mouth and takes their average for the final result. While the OpenCV implementation of Viola-Jones for face and mouth detection in fact helps detecting the mouth within a face, sometimes this method fails to detect yawning since the it does not discriminate a wide open mouth, as in yawning, from a mouth just barely open. In fact, the Viola Jones algorithm is trained to detect the mouth only, not to discriminate between different degrees of mouth openings. Instead of taking their average, we take the biggest candidate as the final detected mouth. This may explain why yawning mouths have a higher chance of being detected as they are normally bigger than a normal mouth.

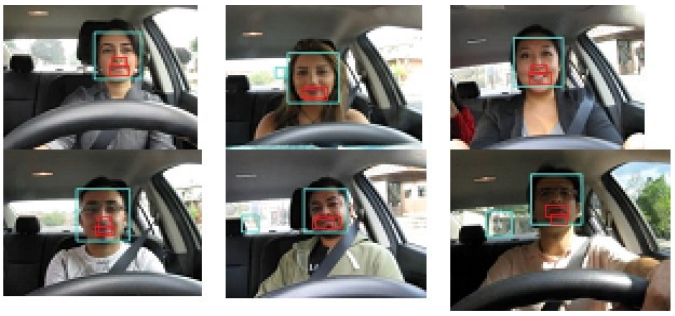

Fig. 12. OpenCV's incorrect multiple mouths and faces detection.

It should also be mentioned that because of using back projection theory in our design for detecting yawning, the false positives are low in our experiments. Since yawning is detected based on a sequence of images and according to a specific temporal relationship as explained in Section III.C, false positives are reduced. However, this is limited to our experiments. In the real world, if a sequence of images looks like yawning, for example, singing a song where the mouth gradually opens and then gradually closes according to the same temporal profile explained in Section III.C, in theory it is possible for this sequence to be incorrectly identified as yawning.

VII. CONCLUSIONS

A computationally lightweight method based on the Viola-Jones theory for face and mouth detection, and a back projection technique designed for yawning detection was proposed in this paper. The proposed system was implemented and tested on the CogniVue APEX embedded smart camera, and the results indicate promising accuracy and reliability. The results of the proposed method are compared with other methods representative of the state of the art, and the experimental results suggest that the proposed method potentially can detect yawning with a higher accuracy (on average).The embedded platform uses a small camera installed under the front mirror or on the dash of a car. The output of the camera is processed in the embedded platform using our

system and the results of face and mouth tracking as well as yawning alert signal can be seen on the monitor. To make the system work on a computationally limited platform, much effort was made in designing and optimizing algorithms and codes to work in real time and without requiring high level hardware platforms. The yawning detection results can be employed for drowsiness monitoring in future work.

REFERENCES

[1] P. Smith et al, "Determining driver visual attention with one camera", IEEE Trans. on Intelligent Transportation Systems, Vol. 4, Issue 4, , pp. 2015-2018, January 2004.

[2] M. Rezaei, and R. Klette, "Look at the Driver, Look at the Road: No Distraction! No Accident!", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colombus, OH, USA, pp. 129 – 136, 23-28 June 2014.

[3] J.R., Treat, "Tri-level study of the causes of traffic accidents," Report No. DOT-HS-034-3-535-77 (TAC), 1977.

[4] S.G. Klauer, T. A. Dingus, V. L. Neale, J.D. Sudweeks, and D.J. Ramsey, "The Impact of Driver Inattention on Near-Crash/Crash Risk: An Analysis Using the 100-Car Naturalistic Driving Study Data," Virginia Tech Transportation Institute, Technical Report # DOT HS 810 594.

[5] O. Tunçer, L. Güvenç, F. Coşkun and E. Karslıgil, "Vision based lane keeping assistance control triggered by a driver inattention monitor," in *IEEE Int'l Conference on Systems Man and Cybernetics (SMC),* Istanbul, 10-13 Oct. 2010.

[6] I. Takahashi et al., "Overcoming Drowsiness by Inducing Cardiorespiratory Phase Synchronization", IEEE Trans. on Intelligent Transportation Systems, Vol. 15, Issue 3, pp. 2015-2018, June 2014.

[7] H.B. Kang, "Various Approaches for Driver and Driving Behavior Monitoring: A Review", *Proc. IEEE International Conference on Computer Vision Workshops (ICCVW)*, Sydney, Australia, pp. 616-623, 2-8 Dec. 2013.

[8] Y. Andreu-Cabedo et al., "Mirror mirror on the wall… An intelligent multisensory mirror for well-being self-assessment", *IEEE Conference on Multimedia and Expo (ICME)*, Turin, Italy, June 29 -July 3 2015.

[9] S.I. Ali et al., "An efficient system to identify user attentiveness based on fatigue detection", *Proc. Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India, pp. 15-19, 1-2 March 2014.

[10] M. Sasaki et al., "Estimation of tongue movement based on suprahyoid muscle activity", *Proc. International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Nagoya, Japan, pp. 433 – 438, 6-9 Nov. 2011.

[11] S. Shirmohammadi and A. Ferrero, "Camera as the Instrument: The Rising Trend of Vision Based Measurement", *IEEE Instrumentation and Measurement Magazine*, Vol. 17, No. 3, pp. 41-47, June 2014.

[12] G. Bradski, A. Kaehler, "Learning OpenCV Computer Vision with the OpenCV library," O'Reilly, Ch. 13. p.p. 509, 2008

[13] C.A.R, Behaine, J. Scharcanski, "Enhancing the Performance of Active Shape Models in Face Recognition Applications," IEEE Transactions on Instrumentation and Measurement, vol. 61, Issue 8, pp. 2330 – 2333, 2012.

[14] G. Betta, D. Capriglione, M. Corvino, C. Liguori, A. Paolillo, "Face Based Recognition Algorithms: A First Step Toward a Metrological Characterization," IEEE Transactions on Instrumentation and Measurement, vol. 62, Issue 5, pp. 1008 – 1016, 2013.

[15] M.S. Hosseini, B.N. Araabi, H. Soltanian-Zadeh, "Pigment Melanin: Pattern for Iris Recognition," IEEE Transactions on Instrumentation and Measurement, vol. 59, Issue: 4, pp. 792 – 804, 2010.

[16] S.S. Beauchemin, M.A. Bauer, T. Kowsari, C. Ji, "Portable and Scalable Vision-Based Vehicular Instrumentation for the Analysis of Driver Intentionality," IEEE Transactions on Instrumentation and Measurement, vol. 61, Issue: 2, pp. 391 – 401, 2012.

[17] S. Abtahi, S. Shirmohammadi, B. Hariri, D. Laroche, and L. Martel, "A Yawning Measurement Method Using Embedded Smart Cameras," *Proc. IEEE Int'l Instrumentation and Measurement Technology Conference*, Minneapolis, USA, May 6-9, 2013.

[18] H.K. Jee, S.U. Jung, J.H. Yoo, "*Liveness detection for embedded face recognition system*," Int. J. of Biomedical Sciences, vol. 1(4), pp. 235-238, 2006

[19] M. Yang, J.E. Crenshaw, B. Augustine, R. Mareachen, Y. Wu, "AdaBoost-based face detection for embedded systems," Computer Vision and Image Understanding, vol. 114, issue 11, pp. 1116-1125, 2010.

[20] C. Bouvier, A. Benoit, A. Caplier, P.Y.Coulon, "Open or Closed Mouth State Detection: Static Supervised Classification Based on Log-polar Signature," Advanced Concepts for Intelligent Vision Systems Juan-les-Pins, France. Springer Berlin / Heidelberg, vol. 5259, pp.1093-1102, 2008.

[21] C.C. Chiang, W.K. Tai, M.T. Yang, Y.T. Huang, C.J. Huang, "A Novel Method for Detecting Lips, Eyes and Faces in Real Time," Real-Time Imaging 9, pp. 277–287, 2003

[22] V. P. Minotto, C.B.O. Lopes, J. Scharcanski, C.R. Jung, B. Lee, "Audiovisual Voice Activity Detection Based on Microphone Arrays and Color," IEEE Journal of Selected Topics in Signal Processing, vol. 7, issue 1, pp. 147-156, June 2014.

[23] R. Medeiros, J. Scharcanski, A. Wong, "Multi-scale Stochastic Color Texture Models for Skin Region Segmentation and Gesture Detection," *IEEE Int'l Conference on Multimedia and Expo (ICME)*, San José, USA, 15-19 July 2013.

[24] A. Bigdeli, Abbas, C. Sim, M. Biglari-Abhari and B. C. Lovell, "Face Detection on Embedded Systems," Proceedings of the 3rd Int'l Conf. on Embedded Software and Systems: Lecture Notes in Computer Science. Korea, pp. 295-308, 14-16 May 2007.

[25] X. Fan, B. Yin, Y. Fun. "Yawning Detection For Monitoring Drive Fatigue." In: Proc. Sixth International Conf. on Machine Learning and Cybernetics, Hong Kong, pp. 664-668, 2007.

[26] T. Azim, M.A. Jaffar, A.M. Mirza. "Automatic Fatigue Detection of Drivers through Pupil Detection and Yawning Analysis," In: *Proc. Fourth Int'l Conf. on Innovative Computing, Information and Control*, pp. 441-445, 2009.

[27] L. Li, Y. Chen, Z. Li, "Yawning Detection for Monitoring Driver Fatigue Based on Two Cameras," *Proc. 12th Int. IEEE Conf. on Intelligent Transportation Systems*, St. Louis, MO, USA, pp. 12-17, 2009.

[28] Y. Ying, S. Jing, Z. Wei, "The Monitoring Method of Driver's Fatigue Based on Neural Network," *Proc. International Conf. on Mechatronics and Automation*, pp. 3555-3559, 2007.

[29] M. Saradadevi, P. Bajaj, "Driver Fatigue Detection Using Mouth and Yawning Analysis," IJCSNS Int'l Journal of Computer Science and Network Security, vol. 8, no. 6, pp. 183-188, 2008.

[30] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlett and J. Movella "Drowsy Driver Detection Through Facial Movement Analysis", ICCV *Workshop on Human Computer Interaction*, 2007.

[31] K. Barry, "Yawn if you Dare. Your Car is Watching You," Wired Magazine, Autopia Section, July 30, 2009.

[32] S. Abtahi, B. Hariri, and S. Shirmohammadi, "Driver Drowsiness Monitoring Based on Yawning Detection", *Proc. IEEE International Instrumentation and Measurement Technology Conference*, Binjiang, Hangzhou, China, May 10-12 2011.

[33] P. Viola and M. Jones, "Robust real-time face detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137-154, 2001

[34] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, "YawDD: A Yawning Detection Dataset," *Proc. ACM Multimedia Systems,* Singapore, pp. 24-28, 2014.

[35] H. Zhou, Q. Tang, L. Yang, Y. Yan, G. Lu, K. Cen, "Support vector machine based online coal identification through advanced flame monitoring," Fuel 117, pp. 944-951, 2014.

APPENDIX: COMPUTATIONAL COMPLEXITY

A. Computational complexity estimate for [20]

This method requires the mouth image as an input, and we used the Viola-Jones algorithm to detect the mouth image as the input for this method. Our proposed method, in contrast, finds face, mouth and its state (yawning/Not-yawning) automatically.

Computational complexity of the training stage:

As this method needs the mouth image for training the SVM, the mouth should be provided as an input. The procedure after finding the mouth has the following complexity.

Computational complexity of the retina filtering stage:
Computational complexity = $O(S'K_sN_s)$
S'=size of the mouth block (number of pixels in the mouth bounding box)
K_s= kernel size
N_s=number of the mouth samples for training the SVM
Computational complexity of the Log-polar signature:
Computational complexity = $O(S'N_s)$

Computational complexity of the PCA dimensionality reduction:
Computational complexity = = $O(N_s(P^2N_s + P^3))$
P= Number of feature points which are generated by the log-polar signature

Computational complexity of the SVM training stage:
Computational complexity = $O(\max(N_s, d)\min(N_s, d)^2)$
d = number of features (dimensions of each mouth sample)

Overall computational complexity of the training stage:
$O(S'K_SN_S + S'N_S + N_S(P^2N + P^3) + \max(N_S, d)\min(N_S, d)^2)$
$= \boldsymbol{O(S'K_SN_S + N_S(P^2N + P^3) + \max(N_S, d)\min(N_S, d)^2)}$
Note 1: $S'N_S$ is ignored because the term $S'K_SN_S$ is greater than $S'N_S$.
Note 2: This estimate assumes that the mouth is given as input, otherwise the "computational complexity of mouth detection training stage" given for our method must be added to this estimate.

Computational complexity of the testing stage:
As this method needs the mouth image for training the SVM, the mouth should be provided as an input. The procedure after finding the mouth has the following complexity.

Computational complexity of the retina filtering stage:
Computational complexity = $O(S'K_s)$
S' =Number of pixels in the bounding box of the mouth
K_S= kernel size

Computational complexity of the Log-polar signature:
Computational complexity = $O(s')$

Computational complexity of the PCA dimensionality reduction:
Computational complexity = $O(P^2S' + P^3)$

Computational complexity of the SVM testing stage:
Computational complexity = $O(S_V)$
S_V = Number of Support vectors
Note: The classifier is expressed in terms of the number of support vectors, and the classification is linear in the number of such vectors.

Overall computational complexity of the testing stage:
$O(S'K_S + S' + P^2S' + P^3 + S_V) = \boldsymbol{O(S'K_S + P^2S' + P^3 + S_V)}$

Note 1: S' is ignored because term $S'K_S$ is greater than S'.

Note 2: This estimate assumes that the mouth is given as input, otherwise the "computational complexity of mouth detection training stage" given for our method must be added to this estimate.

B. Computational complexity estimate for [21]

The method described in [21] extracts skin pixels based on rules derived from a quadratic polynomial model, and this polynomial model is also applicable to the extraction of lips pixels. The extraction of lips and mouth pixels is followed by the extraction of the eyes components using histogram equalization of grayscale image and thresholding, after which the falsely extracted eyes components are removed by verifying spatial and geometrical relationships between eye components and the lips/mouth.

This method does not involve any training, so the computational complexities given below are the testing complexities.

Computational complexity of the testing stage:
Computational complexity of the skin/face pixels and lips pixels extraction stage:
Computational complexity = $O(IK)$
I = number of pixels in the input image
K = number of operations performed upon each pixel in the input image, which in our case correspond to quadratic polynomial model verification for upper and lower boundaries of skin and lips locations, for extraction of face and lips pixels.

Computational complexity of the eyes components extraction:
Computational complexity = $O(I)$
I = size of the input image
Note: A thresholding operation is performed on the histogram-equalized grayscale image to extract the eye components, with the computational complexity given above.

Computational complexity of the geometric relationship verification stage:
Computational complexity = $O(GC)$
G = number of operations performed for the verification of the geometrical and spatial relationships between the facial components (eyes and lips/mouth)
C = number of lips/mouth components found in the input image

Computational complexity of the yawning detection stage:
Computational complexity = $O(S')$
S' =Number of pixels in the bounding box of mouth

Overall computational complexity of the testing stage:
$O(IK + I + GC + S') = \boldsymbol{O(IK + GC + S')}$

Note: I is ignored, because IK is larger than I.