Design of an Attention Detection System on the Zynq-7000 SoC

Fynn Schwiegelshohn, Michael Hübner
Embedded Systems for Information Technology
Ruhr-Universität-Bochum
Bochum, Nord-Rhein-Westfalen
Email: Fynn.Schwiegelshohn@rub.de; Michael.Huebner@rub.de

Abstract—**In this paper, we introduce a prototype attention detection system for automotive drivers. The driver is monitored through a Microsoft Kinect camera which provides RGB, depth, and infrared images in order to cover situations in which normal cameras might not achieve good results. The Kinect is connected to a Xilinx ZedBoard wich uses a Zynq-7000 SoC as processing platform. The attention detection system is running on the ARM Cortex-A9 dual core processor of the Zynq-7000 SoC. The system needs to recognize the drivers face and eyes in order to determine his state of attention. If the driver is classified as being attentive, no warning is generated. If the driver is classified as being inattentive, the system will generate a warning. Several algorithmic optimizations have been implemented in order to increase performance of this solution. In order to simulate a realistic driving environment, we have connected the Xilinx ZedBoard with a car simulator. This provides us with the necessary real world data to validate our system design. When our detection system classifies a driver as distracted or drowsy, it will send a warning message to the car simulator. The results show that the system performs satisfactorily when a face is detected. However, if no face is detected, the frame rate drops below an acceptable level.**

Keywords: Zynq SoC; Advanced Driver Assistance Systems; Attention detection; Computer Vision; Kinect sensor

I. Introduction

Traveling with motorized vehicles has become an indispensable part of our society. Almost every person has to participate in traffic on a regular basis. Maintaining a high attentiveness for traffic can be challenging for the driver since a lot of distractions are available such as using mobile phones while driving. But not only distractions can result into potentially dangerous situations. Several National Highway Traffic Administrations have published surveys about severe traffic accidents in context with drowsy driving [1]. Drowsiness occurs especially during long monotonous routes where not much driving input is required. Therefore, safety in automotive systems is a relevant topic and receives a lot of interest from the industry as well as from academia. Advanced driver assistance systems (ADAS) aim to reduce the risk of severe traffic accidents by monitoring the behavior of the driver or by detecting potential dangerous situations and initiate countermeasures. One such ADAS is the adaptive cruise control (ACC) [2]. It monitors the distance to the vehicle in front and automatically reduces its own speed if a certain distance threshold is underrun. ACC systems prevent rear-end collisions when the driver does not react fast enough or fails to assess the situation correctly. Most ADAS only react to actions the vehicle takes and do not monitor the driver thus preventing dangerous situations from happening. Therefore, several automotive companies developed systems which directly monitor the driver. In our approach, we use the Microsoft Kinect camera for image acquisition. The Kinect is a flexible off-the-shelf product which can provide infrared, RGB, and depth images. We connect it to a Xilinx ZedBoard with a Zynq7000 System on Chip (SoC) which runs a Linux operating system. Currently, not many solutions exist to integrate the Kinect seamlessly with an FPGA. This is the first step to integrate the Kinect camera in such a system. Future work will tackle the communication between the Kinect and the programmable logic of the Zynq7000 SoC. The attention detection algorithms used in this approach are implemented in software. Later, most of the computer vision algorithms will be migrated to hardware in order to gain a performance boost and to determine the threshold when the communication overhead between processing system and programmable logic is overcome. For evaluation purposes the ZedBoard can communicate via Ethernet with a driving simulator which then displays a warning message on its screen if the attention detection system classifies the driver as inattentive.

In this paper, we first review existing work in the field of advanced driver attention systems in section II before presenting our attention detection approach in section III. Afterwards, section IV describes the implementation of the algorithm on our target platform. Results of the proposed method are given in section V. This paper closes by discussing future work and drawing conclusions in section VI.

II. Previous Work

Several driver assistance systems have been proposed to monitor the attention of drivers. One approach is to monitor the driver indirectly via his driving input such as Mercedes Benz driver attention detection system ”Attention Assist” [3]. It works with several different sensors. The sensor data is used to create a driving profile which the system references to determine if the driver is still attentive or not. If the system detects greater deviations from the established driving profile, a warning will be issued. Downside of this system is that an initial setup time during every ride is required to create a driving profile. Thus the system can only detect longterm drowsiness and not short periods of inattentiveness. Additionally, this system only works between 80km/h and 180km/h. Another method is to determine the driver's state with two cameras using different evaluation methods. T. Victor and A. Zelinsky present a method which uses stereo vision algorithms to match features of the left and right image to extract the 3D position of each feature [4]. The pose of the head is determined via a least squares optimization and the eyes are located by searching for iris features. The eye processing method then combines the eye-gaze vectors for each eye with the head pose to calculate the eye-gaze direction. The presented results are very impressive. However, manual initialization of the feature points is required which makes the system cumbersome for automotive vehicles. Toyota pursues a different approach with their ”Driver Monitoring System” [5]. Their system uses two cameras to track the eyes and face of the driver and issues a warning if the driver is detected to be inattentive. The first camera generates a low resolution image to detect the face of the driver and determine its position in the the car. After this initial detection, image processing algorithms are used on the high resolution image of the second camera to detect facial features. Smith et al. introduce a system for analyzing human driver visual attention [6]. It collects data with a single camera which is placed on a car dashboard. The main indications for an attentive state are rotation of the head and eye blinking. If the rotation of the head deviates to far from the frontal direction, the system generates a warning. The same procedure is followed when the systems detects closed eyes for a longer period of time or when the eye blinking rate indicates drowsiness of the driver. Our system also follows the single camera approach but we use off-the-shelf products like the Microsoft Kinect. With the Kinect we have access to RGB, depth and infrared image data which we use to develop our attention detection system. With these different types of image data available from one hardware component, we can increase the accuracy of our detection algorithms without worrying about synchronization issues. Additionally, our target platform is a Zynq-7000 SoC which enables us to easily migrate from a software only solution to a hardware/software codesign in future works.

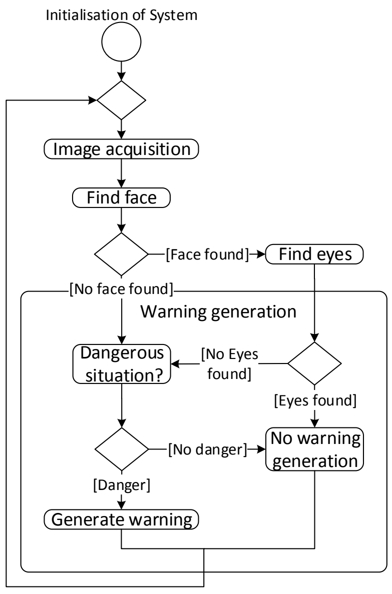
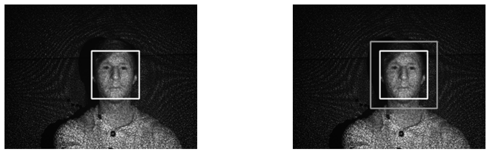


Fig. 1. The flow diagram of the attention detection algorithm



(a) Facial frame (b) Increased facial frame

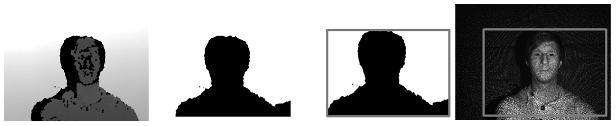
Fig. 2. The simple face tracking method with an increased region of interest to detect small movements between frames

III. Attention Detection Algorithms

The setup for a driver attention detection system is generally well known. The camera will be placed in front of the driver so that the drivers side will be fully captured by the camera. Our initial attention detection system is comprised of three different modules. The complete system is detailed in Fig. 1. The first module acquires the various images from the Kinect camera and sends them to the face detection module. The next module detects the face in the image and either activates the eye detection module or the module which analyzes the data generated by the detection modules and decides if a warning should be generated or not.

A. Face detection

For face detection, we use the algorithm proposed by P. Viola and M. Jones [7]. Their algorithm employs cascaded rectangular Haar-like features and accelerates the reduction and synthesis of these features with the algorithm ”AdaBoost”. We want to optimize our system for runtime performance thus reducing the processing time of the algorithms on the ZedBoard as far as possible. Detecting facial features in an image takes longer with increasing image size. Therefore, we try to reduce the input image size before initiating the face detection algorithm. If a face is already detected in the image, the position and size of the rectangle surrounding the face is stored, as seen in Fig. 2a. In all of the following images, we will first initiate the face detection algorithm in the image region where the former face was detected. Depending on the size of the detected face, we are able to reduce the image size considerably. However, if we only use the frame size in which the face is detected, small movements of the driver will result in no face being found and then face detection has to be executed on the whole image again. To compensate small movements of the driver, we therefore increase the size of the rectangle framing the detected face, see Fig. 2b. This results in slightly larger partial images than optimal, but this simple solution greatly increases the detection rate between several images where drivers are moving their head. These optimization steps only work when a face has been detected in the previous image. If no face is detected, the whole image has to be processed. Since the Kinect provides us with depth images, we are able to permanently reduce the input image size by employing foreground isolation and thus accelerate facial detection. With the depth image, we are able to divide the image into a foreground and a background region. This is done through binarisation and contour detection. In the depth image, every pixel has a value from 0 to 255 associated with the distance to the Kinect camera. A high value of a pixel in the depth image resembles a larger distance from the



(a) Depth image to binary (b) Foreground projection to infrared image

Fig. 3. Cropping the infrared image through foreground isolation

(a) Eyes open (b) Eyes closed

Fig. 4. Eye detection in the cropped infrared image

Kinect camera. Based on our experimental setup, we define a threshold to divide the image into foreground and background pixel. All pixel values lower or equal than the threshold will be assigned the value 0, all pixel values higher than the threshold will be assigned the value 255. The conversion into the binary depth image is shown in Fig. 3a. We use this binary depth image to detect the contours of the largest black pixel region. This is done with the contour detection method from S. Suzuki and K. Abe [8]. The binary image will be searched line by line for a long sequence of ”white” pixel followed by a ”black” pixel. This type of sequence marks the start of the contour detection. Starting from this pixel, all neighboring pixel will be examined counter clockwise and the first detected neighboring black pixel will be the next starting point in the following iteration. This is done until the algorithm reaches the initial starting point again. Afterwards, the algorithm will continue its search for the sequence in the same line from where it detected the starting point. The largest black contour is then classified as foreground and its size will be given to the facial detection system in order to incorporate this data to reduce the infrared image size. As seen in Fig. 3b, the contour region is projected onto the infrared image and defines a region of interest. When initializing the system, we do not have an initial facial region with which the image size can be reduced. Therefore, whenever no initial information about the position of the face in the image exists, we only search for faces in the foreground region of the infrared image. Due to this method we never actually process the whole image size of 640×480 pixel.

B. Eye detection

When the position of the face in the image is known, the eye detection module is executed. For eye detection, we also use the algorithm from P. Viola and M. Jones, with Haar-features that are trained to detect eyes in images. The eye detection will only take place in the small image snippet which is obtained through the face detection step. We can further reduce this cropped image by assuming that the eyes are only present in the upper half of the image snippet. The resulting image snippets with open and closed eyes are shown in Fig. 4. As seen in Fig. 4a, only open eyes are detected. To increase detection robustness, the algorithm only searches for one open eye. If no eyes have been detected by the algorithm as seen in Fig. 4b, the system classifies the detected face as inattentive due to assuming that the eyes are closed.

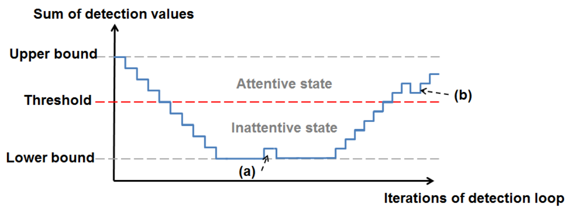


Fig. 5. Example behavior of a hysteresis classifier

C. Warning generation

An attention detection system must be able to determine if a given situation is dangerous or not. For instance, when the driver is looking over the shoulder to determine if another vehicle is present in his blind spot, the attention detection system should not generate a warning. False positive and false negative detections can influence the attention classification in a wrong manner, thus giving warning signals when no danger is present and failing to warn the driver in a critical situation. We employ a simple hysteresis classifier, which determines if a warning should be generated or not. This approach simply counts the amount of positive and negative detections during runtime. We define a maximum S_{max}, a minimum S_{min}, and a threshold value $S_{threshold}$. S_{max} determines the highest number the classifier can achieve with a long sequence of positive detections. S_{min} is the lowest number the classifier can achieve with a long sequence of negative detections. These three values are determined empirically through a series of test cases which aim to reduce the amount of false positive and false negative detections in the system. This behavior is described in equation (1).

$$S_t = \begin{cases} S_{max} & , \quad S_{t-1} + x_d \geq S_{max} \\ S_{min} & , \quad S_{t-1} + x_d \leq S_{min} \\ S_{t-1} + x_d & , \quad else \end{cases} \tag{1}$$

S_t and S_{t-1} resemble the hysteresis values of the current and former time step respectively, x_d is the result of the face and eye detection which can be either 1 or -1. When the current classifier value S_t falls below or rises above $S_{threshold}$, the classifier will change its state to either ”inattentive” or ”attentive”. This is described in equation (2).

$$state = \begin{cases} attentive & , \quad S_t > S_{threshold} \\ inattentive & , \quad S_t < S_{threshold} \\ former\ state & , \quad S_t = S_{threshold} \end{cases} \tag{2}$$

When S_t reaches the threshold, the state does not immediately change. It waits for the next detection. If this detection leads to a value which passes the threshold, the state shifts. This avoids constant state changes when the system alternately detects a face and fails to detect one, since at least two consecutive detections must give the same results. An example process of the hysteresis classifier is given in Fig. 5. Here, the classifier starts with a fully saturated value for the ”attentive” state. The system then fails to detect faces in several consecutive images before a face can be detected again. The two interesting points in this figure are a false positive detection marked as (a) and a false negative detection marked as (b). It can be seen that small amounts of false detections do not impact the classifier in its performance. However, one issue with this solution is the delay in warning generation since the system needs

$S_{max} - S_{threshold}$ face detections in a worst case scenario to change the state. The parameters of the classifier have to be chosen carefully regarding the frame rate and the processing speed of the ARM Cortex A9 Dual Cores. Our solution does not require any calibration steps beforehand. It is basically a plug an play setup. The only component which has to be adapted if the processing speed severely changes is the warning generation since the performance of the hysteresis approach depends heavily on the system processing time. The proposed solution is very flexible regarding its application because of the aforementioned lack of calibration. With the help of the Kinect camera, we are able to adapt to different lighting situations and either use the RGB image or the infrared image. The depth image also provides further information regarding the desired region of interest. However, a drawback of this solution clearly is the warning generation with the hysteresis approach. Depending on the processing time of the detection algorithms, the parameters for S_{min}, $S_{threshold}$, and S_{max} have to be determined empirically. Therefore in future work, a different classifier for the warning generation should be considered.

IV. Implementation

As already mentioned in section I, our system is comprised of a Microsoft Kinect camera, a Xilinx ZedBoard and a driving simulator. For the current implementation, several equally qualified processors exist. However, the Zynq7000 SoC on the ZedBoard enables hardware software codesign approaches in the future through it programmable logic area and the processing system with the ARM Cortex A9 Dual Cores. We are therefore able to transfer the image processing methods to the programmable logic and gain a significant speedup which is not possible on a many other platforms. The ZedBoard is connected via an USB port to the Kinect camera and via an Ethernet port to the driving simulator. Therefore, we need to activate the USB0 and Enet0 peripheral in our ZedBoard design, see Fig. 6. In order to access the image data provided by the Kinect it is mandatory to run a Linux or Windows operating system on the target platform. This is due to the drivers which enable easy access to several features of the Kinect, such as tilting the motor and acquiring infrared, depth or RGB images. We decided to run the embedded Linux operating system Linaro because of its well documented features and its similarity to desktop versions of Linux. Linaro will provide the attention detection system with the required Kinect drivers, the opencv libraries and the communication interface with the driving simulator. The operating system will be stored on a SD card. Therefore the SD0 peripheral of the ZedBoard must be configured properly. The hardware configuration of the ZedBoard is shown in Fig. 6. For monitoring purposes and easy access we activate the HDMI graphical output from the ZedBoard. The HDMI interface is only available through the programmable logic. Therefore, by inserting the ”axi_hdmi_tx_16b” block as well as the ”VDMA” the processing system is able to communicate with the HDMI interface.

V. Results

We will evaluate our system with several test cases. First we compare the different optimized attention detection algorithms regarding their execution time and performance. This is

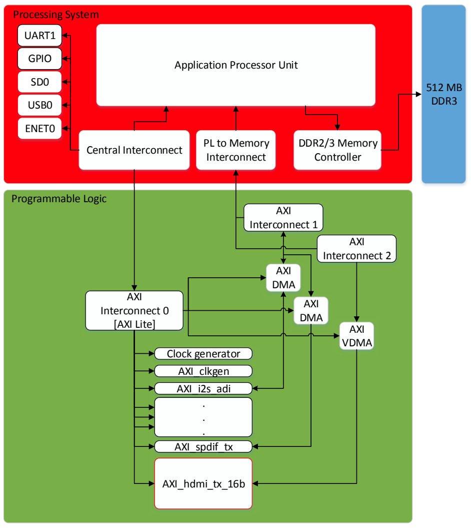


Fig. 6. Hardware configuration of the ZedBoard for the attention detection system

TABLE I. System parameters for evaluation

Parameter	Value	Description
Background threshold	1.45m	Every pixel beyond this distance from the sensor will be classified as background
Frame scale factor	1.2	This is the extended region of interest for the next iteration of face detection

TABLE II. Movement of driver in the first test case

Frames	Description
0-	Start of test scenario
88-106	Head movement towards border of frame (right)
169-191	Head movement towards border of frame (left)
269-351	Head turns sideways (right)
452-532	Head turns sideways (left)
680-830	Person stands up
-1000	End of test scenario

done on a desktop PC running an Ubuntu 12.04.3 LTS Linux distribution with an Intel Core i5 3.4GHz as processor and 8GB RAM. The final speedup results from the PC implementation can differ from the ARM implementation, but an assertion on the performance gain can still be made. We have to define two parameters throughout the whole evaluation in order to have representative results. The parameter values are presented in Table I. Initial tests showed that the maximum distance from the camera within the driving simulator can be 1.45m. In future work, we will aim to set the background threshold adaptively in order to better react to a changing environment. The frame scale factor has to be based on the frame rate of the complete system. If the frame rate is low, then the tracked head can cover a greater distance between two frames and thus leave the prior defined region of interest. Preliminary tests on the target platform showed that a frame scale factor of 1.2 yields satisfactory face tracking results. The test case was executed in the driving simulator. The Kinect camera is placed on the dashboard of the driving simulator and the driver goes through a series of movements which might occur during driving. The movement description of the driver is depicted in Table II. At first the driver keeps his head frontal to the road in the defined ”target position” pose. The driver always reverts to the ”target position” pose between the states mentioned in Table II. After 88 frames, the head

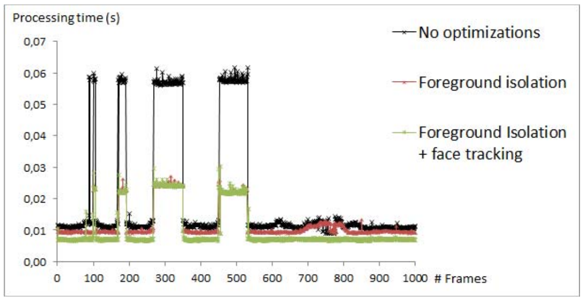


Fig. 7. Processing times per frame for each algorithm optimization step

TABLE III. RESULTS OF THE FIRST TEST CASE

	No optimization	Foreground isolation	Foreground isolation + face tracking
Complete processing time (ms)	20498	12545	10258
Mean processing time per frame with face detected (ms)	11	10	7
Mean processing time per frame with no face detected (ms)	58	23	23
Performance gain with face detected	100%	91%	64%
Performance gain with no face detected	100%	40%	40%

moves towards the border of the image to determine if face tracking works correctly and if face detection also performs well near the image border. Afterwards, the driver simulates inattentive behavior by moving his head sideways. In the last test case, the driver stands up to test the performance of face tracking and face detection in vertical direction as well. After the last test case, the driver reverts to and stays in the ”target position” pose. We compare the face detection system with no optimizations, with foreground isolation, and with foreground isolation and face tracking to demonstrate to effectiveness of these optimizations. These test cases were all executed with different lighting conditions. Our system performed satisfactory in all test cases except in complete darkness or extreme overexposure to light. The processing times for each frame of every optimization step are depicted in Fig. 7. You can clearly see a speedup in processing time when comparing the three attention detection implementations during the ”target position” frames. While moving the head sideways towards the border of the image, the implementation with no optimizations has more problems in finding a face in the image than the other implementations. This attribute does not occur when the head is turned sideways and when the driver stands up in front of the camera. The implementation with foreground isolation and face tracking performs better than the other methods during the period where the driver is standing as the others have small variations in processing time. This variation occurs because the foreground isolation returns a cropped image which is almost the size of the full image due to the person standing in front of the camera. If we also use face tracking, the foreground isolation will not be used and we therefore have a constant processing time throughout the time the driver is standing. Table III shows all results from the measurements. Compared to the implementation with no optimizations, the implementation with face tracking only takes roughly 50% of the time and the implementation only using foreground isolation takes 60% of the time. This is also

TABLE IV. MOVEMENT OF DRIVER IN THE SECOND TEST CASE

Frames	Description
0-30	Head movement towards border of frame (right and left)
42-63	Head turns sideways (right)
105-125	Head turns sideways (left)
168-194	Eyes closed
241-265	Eyes closed
-293	End of test scenario

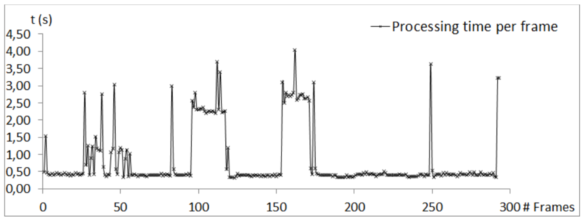


Fig. 8. Processing times per frame for the attention detection system on the target platform (ZedBoard)

apparent when analyzing the mean processing time per frame. If a face is detected the system clearly benefits from the face tracking method, since the region of interest is much smaller than the region gained from foreground isolation. Foreground isolation does not have a much better performance compared to no optimization when a face is detected. However, if no face is being detected, the performance greatly increases with foreground isolation. The mean processing time per frame is reduced to 40% of its original value. This is also true for the face tracking implementation since the face tracking algorithm cannot crop the image any further when no face is being detected. The performance benefit from foreground isolation and face tracking is noticeable. Therefore, we implement the attention detection system with foreground isolation and face tracking on the ZedBoard and perform further tests on the target platform.

The test case on the target platform is described in Table IV. Just like in Table II, the driver always reverts to the ”target position” pose in all frames which are not mentioned in Table IV. This test case consists of the same movements as the first test case with the addition of two segments where the driver closes his eyes. All these tests were performed with different lighting conditions and different persons to ensure reliable results. The driver starts in the ”target position” pose and immediately moves his head towards the image borders. This is done to determine the hysteresis behavior when the system does not have much time to settle into a stable state. Then, the driver turns his head sideways and afterwards assumes the ”target position” pose. While in the ”target position” pose, the driver will close his eyes in order to test the eye detection and the respective hysteresis warning system. However, in order to test the hysteresis warning system, the hysteresis parameters have to be adjusted to the processing speed of the target platform. Therefore, we first analyze our system with focus on the processing time per frame. The results are depicted in Fig. 8. Compared to the test case on the desktop PC, the processing time is much slower with 0.46s when a face is detected and even 2.61s when no face is detected. Otherwise, the system shows the same behavior when executed on the target platform compared to the desktop PC. When the driver moves his head towards the image border, the processing time increases, partly because the face detection method needs to

TABLE V. HYSTERESIS PARAMETERS FOR TARGET PLATFORM

Hysteresis:	Face detection	Eye detection
S_{max}	3	5
$S_{threshold}$	2	3
S_{min}	1	1

TABLE VI. REACTION TIMES OF THE HYSTERESIS CLASSIFIER

State transition	Face detection	Eye detection
inattentive$\rightarrow$attentive	0.92s-3.53s	1.38s-1.84s
attentive$\rightarrow$inattentive	5.22s-5.68s	1.38s-1.84s

adapt to the new situation and partly because the system loses track of the face in the image. When the head moves sideways, the system takes longer to process the image. This is because it does not detect a frontal face anymore and therefore considers the driver to be in a potential inattentive state. When the driver only closes his eyes and otherwise stays in the ”target position” pose, the system does not require noticeably more processing time when the eyes are detected or not. The small peaks in processing time at frame 249, 282, and 286 are unusual since no action was taken which could prompt the system to such an increase in processing time. Further investigations into this matter revealed that the origin for these peaks lies in the image acquisition from the Kinect camera. Just before these peaks occur, the Kinect camera sends a noisy image in which no face detection is possible. The source for this problem can either be the Kinect camera as of itself or the USB communication with the ZedBoard. We assume that the communication and memory storage of the ZedBoard are the reasons the peaks. The attention detection system fails to locate a face within the cropped image and therefore starts to search for facial features throughout the whole image. We could find a solution to this this behavior. Tests with several different environmental parameters such as different lighting, test person, or distance to the Kinect camera could not reproduce the same results. However, the increase in processing time consistently occurred in this test scenario throughout all iterations. With the acquired timing information from this test case, we can now determine the hysteresis values for our final attention detection system. The parameters are shown in Table V. The hysteresis of the eye detection method can cover a larger interval since it only is used when a face is detected and therefore the framerate is sufficiently high for the hysteresis to react faster to a changing attentive state. The face detection method however, has to cope with larger processing times when no face is being detected. The system needs to be able to generate warnings faster when the drivers state of attention changes. Therefore, we decided to use the smallest hysteresis parameters for the face detection method. Through the hysteresis classifier we are able to reduce the amount of false positive and false negative detections to a minimum. The drawback from this classifier is that the reaction time to generate warnings and to recover from a warning state gets delayed. The range of reaction times for face and eye detection is depicted in Table VI. The reaction time of the eye detection hysteresis is constant, although it still takes a large amount of time. This is due to the initial face detection beforehand. The range of the timings can be explained by the current value of the hysteresis parameter S_t. If S_t lies near the threshold, the reaction time towards a changing situation is faster than if S_t is saturated . The state transition timings of the face detection from attentive$\rightarrow$inattentive are very high due to the higher processing time for an image in which no face is detected.

Our attention detection system robustly detects faces and eyes and classifies the drivers state of attention. The optimizations have shown to improve system performance. However, the processing time of an image when no face is detected is still too slow for satisfactory real-time performance. The next steps will be to analyze our system, implement foreground isolation and face detection in hardware on the Zynq-7000 SoC to increase performance even further.

VI. CONCLUSION

In this paper, an attention detection system was implemented on a ZedBoard with a Zynq-7000 SoC and evaluated with several test cases. The system uses a Kinect camera from Microsoft for depth and infrared image acquisition. In order to access the Kinect camera we used the operating system Linaro. Since resources on this embedded platform are limited, we focused on optimizing the base detection algorithms in a way to reduce the overall computation overhead with good results. The final attention detection system was then evaluated on the target platform regarding processing time of each frame and reaction time to changing states of driver attentiveness. While the warning generation of the eye detection method performs satisfactory, the warning generation of the face detection detection method suffers from a long delay. We plan to implement the image processing algorithms on to the programmable logic of the Zynq-7000 SoC. Furthermore, our face detection system has problems when detecting faces with glasses. Different detection algorithms exist for this case, so in future we plan to detect if the driver is wearing glasses and then dynamically adapt our hardware image processing to either detect normal faces or detect faces with glasses.

ACKNOWLEDGMENT

The authors would like to thank the company FOERST for giving us the opportunity to evaluate our system on the driving simulator. Special thanks also go to Xilinx and their university program which provided us with the ZedBoard.

REFERENCES

[1] NHTSA, “National motor vehicle crash causation survey: Report to congress,” DOT HS 811 059, National Highway Traffic Safety Administration, Jul. 2008.

[2] W. Chundrik and P. Labuhn, “Adaptive cruise control,” Patent, Oct., 1995, u.S. Patent Number US 5454442 A.

[3] E. Bukman, L. Galley, K.-P. Kuhn, and D. Neumerkel, “Method and computer program for identification of inattentiveness by the driver of a vehicle,” Patent, Jun., 2005, u.S. Patent Number US 20050128092 A1.

[4] T. Victor and A. Zelinsky, “Automating the measurement of driver visual behavior using passive stereo vision,” in *International Conference Series Vision in Vehicles VIV9*, Aug. 2001.

[5] A. Kawakubo, “Driver imaging apparatus and driver imaging method,” Patent, Sep., 2010, u.S. Patent Number US 20100220892 A1.

[6] P. Smith, M. Shah, and N. da Vitoria Lobo, “Determining driver visual attention with one camera,” in *IEEE Transactions on Intelligent Transportation Systems*, Dec. 2003, pp. 205–218.

[7] P. Viola and J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision 57(2)*, pp. 137–154, 2004.

[8] S. Suzuki and K. Abe, “Topological structural analysis of digitized binary images by border following,” in *Computer Vision, Graphics and Image Processing*, 1985, pp. 32–46.