Driver Monitoring implementation in Adaptive AUTOSAR environment

Milan Đokić
RT-RK Institute for Computer Based Systems
Novi Sad, Serbia
milan.djokic@rt-rk.com

Stefan Nićetin
Faculty of Technical Sciences
Novi Sad, Serbia
stefan.nicetin@rt-rk.com

Gordana Velikić
RT-RK Institute for Computer Based Systems
Novi Sad, Serbia
gordana.velikic@rt-rk.com

Tihomir Anđelić
RT-RK Institute for Computer Based Systems
Novi Sad, Serbia
tihomir.andjelic@rt-rk.com

***Abstract*—Advanced Driver Assistance System (ADAS) algorithms are of significant importance in the modern automotive industry. The algorithms have changed the approach to two important issues of the industry: traffic efficiency, and safety. ADAS algorithms have very demanding requirements, such as real time execution and low memory consumption. Thus, one of the main challenges is to satisfy these requirements without compromising the reliability. Although all components that add to the safety are important, one that is most commonly addressed is drowsiness detection, because the drowsiness is at the top causes of traffic accidents. Automotive grade standards have changed periodically to include and tailor recent techniques and models. In this paper, we present driver drowsiness detection solution which is implemented on the Adaptive AUTOSAR platform.**

Keywords—automotive, software, Adaptive AUTOSAR, ADAS, Driver Monitoring, Computer Vision

I. Introduction

Development in automotive industry is constrained by the safety regulations. To respond to automotive grade constraints, AUTOSAR standard has published adaptive environment, which has higher flexibility and scalability compared to classic AUTOSAR.

Applications that are executed on the Adaptive AUTOSAR (AA) platform [1] must follow specific AA guidelines and requirements. The challenges for implementing the requirements in such environment are complex, because multiple applications need to share platform resources and to adhere to high safety levels. Additional difficulty during development is mandatory real time processing and demand for high reliability during execution.

Traffic safety improvements have been the focal point in development of automotive industry. Thus, ADAS are constantly providing new solutions for "old" topics, such as for example, collision detection, road free space detection, road signs recognition, etc. As driver drowsiness often causes traffic accidents, Driver Monitoring (DM) algorithms are always integrated as one of the main use cases to ADAS platforms. DM algorithm implementation includes large and small object detections, i.e. face and eyes. Artificial intelligence is commonly used for object detection, and for the level of eyes openness. The results are further used to determine a driver fatigue level.

II. Related Work

There are few different approaches to detecting a driver's fatigue level. One of them is using sensors attached to the driver, which measures biological parameters that help determine the driver's fatigue level. For example, a biological parameter may be the driver's heart rate [2]. However, approaches that involve biological measurements may be unpractical and cumbersome for the driver due to rigid sensors attached, or data noise which may require complex processing.

The popular approach is using cameras as sensors and vision based algorithms to extract information from acquired signals. For example, processed information from cameras are used to detect existence of driver's attention on the road [3]. As ADAS algorithms, including DM, have real time processing requirements, they have to be highly optimized to use heterogeneous parallel computing systems [4] [5].

There are many implementations of this commonly used algorithm. Each approach is platform specific and with each technology change it has to be optimized for the new features of targeted platform or environment.

This paper presents an implementation which is adopted for the AA environment. The main goal of the solution was to implement a DM algorithm which will run under the AA Execution Manager (EM) [6] and use the AA diagnostics [7] to publish its state. We will present the algorithm structure first, then its integration with the AA environment. Finally we give an evaluation of our solution by measuring common algorithm parameters.

III. Algorithm Structure

Algorithm implementation is divided into several stages. The Algorithm structure is shown Fig. 1.

Cameras positions enable to capture the driver in center of the image. In *Grayscale & crop* phase preprocessing of input images is done. This step removes irrelevant information. The grayscale image is needed by Viola-Jones framework [8] for frontal face and efficient iris detection. Since it is required that the driver is in the center of the image, we can crop the frame by 35% of the frame's width.

Downscale prepares images for face detection phase and helps achieve faster detection with no accuracy loss. The

978-1-5386-6095-9/18/$31.00 ©2018 IEEE

driver's face occupies a large amount of the input frame and we can downscale the image which is used as input to the face detection stage.

Face detection is implemented using the Viola-Jones framework for frontal face detection. The Viola-Jones framework utilizes a characteristics extracted from grayscale image of human face as parameters for classifier training. We used pre trained classifier. The detection phase outputs a group of rectangles located at detected faces. The face that is closest to the image center is selected as the drivers face. The final output is a rectangle positioned at the selected face, or information that driver face is not found. If the face is not detected during driving, the alert is activated.

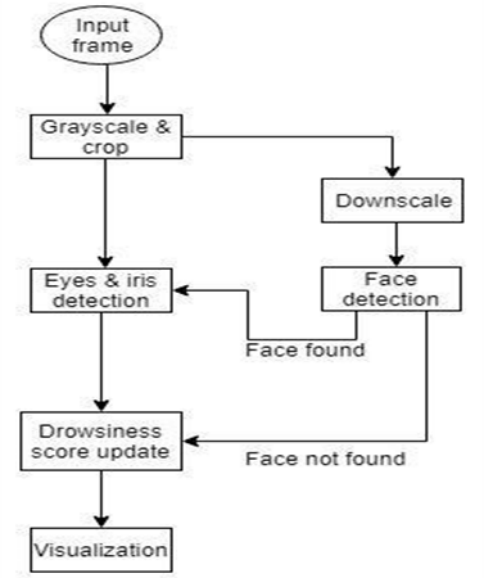

Fig. 1. Algorithm block diagram developed in [3].

Eyes and iris detection is performed only if the driver face is found, otherwise there is no need to run it. After cropping the face rectangle, standard biometric proportions are applied to get rectangles that contain left and right eye. When the eye rectangles are determined, we perform iris detection which is based on a fact that iris is darker than sclera around it. When irises are found, we determine whether the eyes are open or closed. The method is based on assumption that when eye is closed a line of dark pixels are in the eye center, however, if an eye is open such a line is not present. The final outputs are the eye rectangles and information about the eye's aperture.

Score update phase calculates score which is in 0-10 range, with 3 intervals: low, medium, and high. A new score is based on the previous and the detection outputs. If the score value is in the lower interval, the driver is not distracted, otherwise the driver is not paying attention on the road and an alert is activated.

Visualization is done via OpenGL ES. Eye and face rectangles with the drowsiness score bar are drawn over the input frame.

IV. Adaptive AUTOSAR Environment Integration

This is the second part of our solution and explains the key constribution.

Architecture of targeted AA environment in which our DM application has been integrated is shown in Fig. 2. Our DM algorithm is integrated as part of ADAS system on targeted AA platform, together with other ADAS algorithms (Front View, Rear View and Bird View).

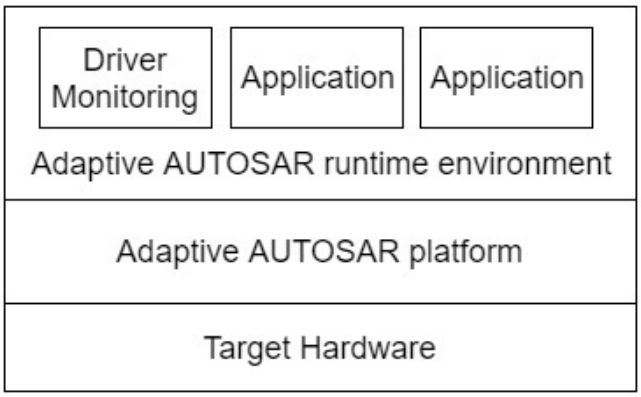

Fig. 2. DM application in target environment

In AA environment application lifecycle management and execution is handled by the EM, while the communication is implemented via the AA communication framework. AA offers a diagnostic framework through which applications can publish events and report their state. For illustration, let assume that driver monitoring algorithm detects that driver is not focusing on the road. To implement ADAS algorithms on the AA platform all steps of the algorithm must be fully integrated in the AA environment. The main challenge was to maintain DM reliability and speed when integrated with the AA environment. DM application shares platform resources - processing power, memory, I/O peripherals, with other applications. It is fundamental to optimize DM application for targeted platform and the AA environment to minimize the impact on functionality by other applications. The DM application runs under the AA EM. The mechanism is based on machine states. For each application is defined in which state it is active. Our platform has predefined machine states: neutral, reverse, and forward, which mimic gear shift. EM is responsible for the machine state management, and for running applications which are relevant to the current state. Hence, integration with EM is based on providing a manifest for our applications with dependencies, where we describe in which states our application is active. Our DM application is active in all target platform states, as driver monitoring is needed in both forward and reverse driving as well as neutral state, i.e. when car is temporarily stopped on the crossroad. Our DM application had dependencies from a camera frame provider service and a Diagnostic Manager (DgM). Application running is performed after running its dependencies. Camera frame provider service is platform specific component which provides input for our algorithm. DM is integrated with AA diagnostics through DgM system service, through which is provided algorithm output

information. DgM system service handles all platform diagnostic information by connecting diagnostic clients with services which provide specific diagnostic information. Every diagnostic service in AA environment has its unique identification number. Diagnostic client applications can request/send data from/to target service through DgM by using service identification number. We implemented DM as AA diagnostic service which provided information about driver drowsiness level. This way, diagnostic client which uses the information can cause appropriate system reaction in case if driver is drowsy (i.e. raise sound alarm or activate brakes to stop the car).

V. EVALUATION

Requirements for the algorithm to perform at maximum efficiency are a light source and a camera placed in front of the driver. Fig. 3. is demonstrating algorithm's functionality.

Fig. 3. top - driver is completely engaged, bottom - eyes closed, drowsiness level bar shown on the left, signal that alert should go off.

Our implementation is tested on an ADAS platform with an AA stack. Note that algorithm shares resources with applications in the AA environment, and as such, it had satisfactory processing speed and high accuracy. Table 1 shows measured algorithm frame rate in two typical scenarios, which differ in drivers frontal face presence in input frame. Algorithm processing frame rate is higher when drivers frontal face is not detected, because in that case eyes detection phase is not performed. Based on the results, the conclusion is that algorithm achieves frame rate of 20 FPS in worst case. Also, we measured influence of shared platform resources in target environment on algorithm frame rate. Comparing to standalone version of algorithm, we had about 1 FPS loss after its integration in target environment (standalone version achieved 21 FPS in worst case).

Scenario	Face not found	Face found
Processing rate	25 FPS	20 FPS

Table 1 – Algorithm processing frame rate

We tested algorithm accuracy in different scenarios and final result is presented via confusion matrix (Table 2). Based on the results, we conclude that measured algorithm accuracy is 95% in average. Missed cases are mostly happening in individual frames. As drowsiness level is calculated based on the results for current frame and previous results, these missed cases are not pushing algorithm results to wrong conclusions.

		Actual case	
		Driver distracted	Driver not distracted
Predicted case	Driver distracted	28	2
	Driver not distracted	1	29

Table 2 – Confusion matrix

The communication with the AA diagnostics is highly efficient - diagnostic applications always have valid and updated results.

VI. CONCLUSION

In this paper we described the implementation of a DM algorithm for the AA environment. As AA is in its early development stage, it is very important to test and confirm functionality of some crucial components like ADAS algorithms and diagnostics of its automotive predecessors (i.e. AUTOSAR classic) in an AA environment. Therefore, the main outcome of this solution is the proof that one of the most important ADAS algorithms is functional in the AA environment. Our algorithm has been integrated with some core components of AA platform such as platform communication mechanism, diagnostics and execution management. Although this presents lightweight integration with AA environment, some AA platform advantages are already noticeable. Our application functionality parameters can be dynamically changed through manifest file, without changing application source code. Also, different diagnostic clients can dynamically connect with our DM service and acquire information about driver drowsiness level.

As AA is in early stage of its development and usage, there is a lot of space for future work on this paper subject. AA opens possibility for remote applications update in system runtime (OTA), which means that algorithm reliability and speed can be dynamically improved.

ACKNOWLEDGMENT

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under Grant TR32029.

REFERENCES

[1] Adaptive AUTOSAR platform [Online], Available: https://www.autosar.org/standards/adaptive-platform/, [Accessed July 23, 2018].

[2] J. Vicente, P. Laguna, A. Bartra, R. Bailón, "Drowsiness detection using heart rate variability" in Medical & Biological Engineering & Computing, June 2016, Volume 54, Issue 6, pp 927–937.

[3] Aleksandra Simić, Ognjen Kocić, Milan Z. Bjelica and Milena Milošević.

"Driver monitoring algorithm for Advanced Driver Assistance Systems", 2016 24th Telecommunications Forum (TELFOR).

[4] Jinglin ZHANG, Jean-Francois NEZAN, Jean-Gabriel COUSIN. "Implementation of Motion Estimation Based On Heterogeneous Parallel Computing System with OpenCL". Jinglin ZHANG, Jean-Francois NEZAN, Jean-Gabriel COUSIN. Universit´e Europ´eenne de Bretagne, France INSA, IETR, UMR CNRS 6164 20, Avenue des Buttes de Coesmes, 35708 RENNES, France.

[5] Nitin Singhal, In Kyu Park, and Sungdae Cho. "Implementation and optimization of image processing algorithms on handheld GPU". Digital Media & Communication R&D Center, SAMSUNG Electronics Co. Ltd., Suwon, Korea. School of Information and Communication Engineering, Inha University, Incheon, Korea.

[6] Specification of Adaptive AUTOSAR Execution Management, release 17.10, October 2017, https://www.autosar.org/fileadmin/user_upload/standards/adaptive/17-10/AUTOSAR_RS_ExecutionManagement.pdf.

[7] Specification of Diagnostics for Adaptive Platform, release 17.10, October 2017, https://www.autosar.org/fileadmin/user_upload/standards/adaptive/17-10/AUTOSAR_SWS_AdaptiveDiagnostics.pdf

[8] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features" in *Computer Vision and Pattern Recognition, 2001. CVPR 2001.*