IET Intelligent Transport Systems

Special Issue: Selected Papers from the 22nd ITS World Congress, Bordeaux 2015

ISSN 1751-956X
Received on 29th January 2016
Revised 1st July 2016
Accepted on 26th August 2016
E-First on 27th September 2016
doi: 10.1049/iet-its.2016.0026
www.ietdl.org

Embedding vision-based advanced driver assistance systems: a survey

Gorka Velez[1] ✉, Oihana Otaegui[1]

[1]Intelligent Transport Systems & Engineering, Vicomtech-IK4, Paseo Mikeletegi 57, San Sebastian, Spain
✉ *E-mail: gvelez@vicomtech.org*

Abstract: Automated driving will have a big impact on society, creating new possibilities for mobility and reducing road accidents. Current developments aim to provide driver assistance in the form of conditional and partial automation. Computer vision, either alone or combined with other technologies such as radar or lidar, is one of the key technologies of advanced driver assistance systems (ADAS). The presence of vision technologies inside the vehicles is expected to grow as the automation levels increase. However, embedding a vision-based driver assistance system supposes a big challenge due to the special features of vision algorithms, the existing constrains and the strict requirements that need to be fulfilled. The aim of this study is to show the current progress and future directions in the field of vision-based embedded ADAS, bridging the gap between theory and practice. The different hardware and software options are reviewed, and design, development and testing considerations are discussed. Additionally, some outstanding challenges are also identified.

1 Introduction

Highly industrialised countries aim to increase mobile efficiency in terms of energy, time and resources as well as to reduce traffic related accidents [1]. Although enormous effort has been done to increase traffic safety, each year more than 1.2 million people still dies in traffic accidents worldwide. Road traffic accidents are the leading cause of death among young people between 15 and 29 years, and cost governments around the 3% of gross domestic product [2]. Modern cars include technology to increase car safety and more generally road safety. This concept is known as advanced driver assistance systems (ADAS). Safety features are designed to avoid collisions and accidents by using technologies that alert the driver of potential dangers or by implementing safeguards and taking over control of the vehicle.

Computer vision, together with radar and lidar, is at the forefront of technologies that enable the evolution of ADAS. Radar offers some advantages, such as long detection range (about 1–200 m), and capability to operate under extreme weather conditions. However, it is vulnerable to false positives, especially around road curves, since it is not able to recognise objects. Camera-based systems also have their own limitations. They are very affected by weather conditions, and they are not as reliable as radar when obtaining depth information. On the other hand, they have a wider field of view, and more importantly, they can recognise and categorise objects. For all these reasons, modern ADAS applications use sensor fusion to combine the strengths of all these technologies. Normally, a radar or lidar sensor is used to detect potential candidates, and then, during a second stage, computer vision is applied to analyse the detected objects. Nevertheless, not all applications need sensor fusion, and some applications such as lane departure warning (LDW) or driver fatigue warning can rely entirely on a camera-based system.

The role of computer vision in understanding and analysing the driving scene is of great importance in order to build more intelligent driver assistance systems. However, the implementation of these computer vision-based applications in a real automotive environment is not straightforward. The vast majority of works of the scientific literature test their driver assistance algorithms on standard PCs. When these algorithms are ported to an embedded device, they see their performance degraded and sometimes they cannot even be implemented. Since there are several requirements and constrains to be taken into account, there is a big gap between what is tested in a standard PC and what finally runs in the embedded platform. Furthermore, there is not a standard hardware and software platform, so different solutions have been proposed by the industry and the scientific community, as it is usual on still non-mature markets.

The purpose of this paper is to present an up-to-date survey about different aspects of embedding vision-based ADAS. Section 2 lists the main requisites of embedded vision. An overview of the different hardware options is presented in Section 3, including a systematic review to study what the most frequent options are. Section 4 reviews the available software platforms and Section 5 discusses the design, development, validation and verification procedures. Section 6 presents a discussion that includes the main technical research challenges that vision-based embedded ADAS face, and finally, Section 7 concludes the paper.

2 Requisites of embedded vision

Embedded vision systems for driver assistance need to fulfil a trade-off between several requirements: dependability, real-time performance, low cost, small size, low power consumption, flexibility and fast time-to-market. This section reviews these requirements in order to better understand the decisions taken during the design stage.

2.1 Dependability

Dependability is defined as the property of a computer system such that reliance can justifiably be placed on the service it delivers [3]. In order to determine the overall dependability of a system, several attributes are assessed. According to [3], these attributes are:

- Availability, which means readiness for usage.
- Reliability, which means continuity of service.
- Safety, which implies the non-occurrence of catastrophic consequences on the environment.
- Confidentiality, which implies the non-occurrence of unauthorised disclosure of information.
- Integrity, which implies the non-occurrence of improper alterations of information.
- Maintainability, which implies the ability to undergo repairs and evolutions.

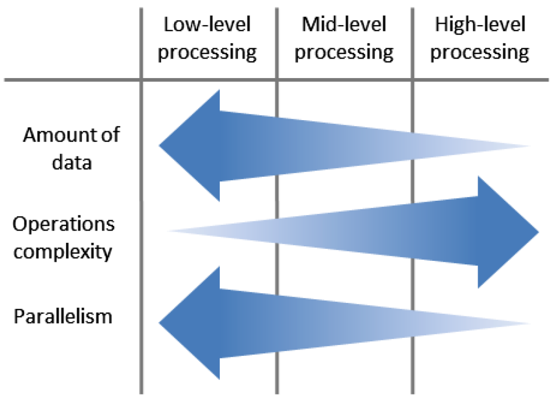


Fig. 1 *Features of each level of processing*

If we associate integrity and availability together with confidentiality, we obtain security, which is also critical.

Impairments or threats are undesired circumstances that can affect negatively to a system's dependability. There are the three main impairment forms:

- Failure: it happens when the delivered service does not fulfil its function.
- Error: it is generated from an invalid state and implies a discrepancy between the intended behaviour of a system and its actual behaviour inside the system boundary. An error may not necessarily cause a failure. The system can response to an error using for instance an exception handling mechanism, and continue operating fulfilling the functional specifications.
- Fault: it is the adjudged or hypothesised cause of an error. The presence of a fault in a system may or may not lead to a failure.

Faults, errors and failures are therefore related terms, which operate according to a mechanism known as fault–error–failure chain [4]. This mechanism works as follows. When a fault is activated it can generate an error. If the error is generated, it creates an invalid state than can lead to a failure, which is a deviation from the specified behaviour of the system that is observable by a user.

In computer vision systems, failures are generally classified as false positives or false negatives. False positives can distract or confuse the driver, or even create dangerous situations. On the other hand, if the system does not alarm whenever is necessary, its utility decreases and it can create a feeling of false safety, which is also dangerous.

2.2 Real-time performance

The system not only needs a robust algorithm, but it also needs to run it fast enough to assist the driver in time. Normally, the required real-time frame rate is between 15 and 30 frames/s.

Obtaining a real-time performance on embedded vision is very challenging, as there is no hardware architecture that meets perfectly the necessities of each processing level. Three different processing levels can be found in computer vision applications: low level, mid-level and high level [5]. Low-level processing is characterised by repetitive operations at pixel level. Typical examples are simple filtering operations such as edge detection or noise reduction. This processing is better served using single instruction on multiple data (SIMD) architectures. The following processing stage, mid-level, is focused on certain regions of interest that meet particular classification criteria. This processing level includes operations such as feature extraction, segmentation, object classification or optical flow. This part of the algorithm has higher complexity than simple filtering and can only be parallelised to some extent. Finally, high-level processing is responsible for decision making, where sequential processing fits better. Fig. 1 summarises the features of each level of processing (adapted from [6]).

2.3 Low cost

As explained before, due to the highly competitive market, the developed embedded device should have a low economical cost. An automotive optical camera costs around 150 USD [7], which is a significant cost. Therefore, it is necessary to minimise product development cost as well as use economical hardware for running the vision algorithms.

2.4 Spatial constraints

There is not much space inside a vehicle to install a camera-based system without affecting to the field of view of the driver. Furthermore, electronic components are very sensitive to temperature and vibrations, so their location inside the vehicle needs to be chosen carefully. A small sized device would facilitate a lot its integration.

2.5 Low power consumption

Power consumption is an important matter in any embedded system, but it is especially relevant in automotive applications, where the energy efficiency is one of the most valuable features of a car. A power consumption of <3 W can be considered satisfactory [8].

2.6 Flexibility

The flexibility of the whole system is an important issue to take into account during architecture design. A flexible ADAS implementation should be able to be updated easily in order to fix detected bugs. Otherwise, an entire hardware replacement would be necessary, which implies higher maintenance costs. Flexibility can be achieved by means of software or reconfigurable hardware.

2.7 Short time to market

The designed ADAS application should reach market fast, so it is necessary to choose architectures that enable rapid development cycles, which also implies lower development costs. However, this requirement often clashes with the requirement of dependability. In order to assure a dependable system, rigorous development procedures need to be followed, what makes it difficult to obtain a short time to market.

3 Hardware

3.1 Overview

The explosion of modern driver assistance technologies started with the first Defense Advanced Research Projects Agency (DARPA) grand challenges, which made these technologies more visible to the general public. The last of these challenges focusing on autonomous driving was the urban challenge that took place in 2007 [9]. In that challenge, most of the teams used desktop computers or small clusters to run their computing demanding algorithms. Those demonstrators were only a proof of concept, so it was not necessary to embed them.

However, a hardware product that is installed inside a vehicle must be embedded and needs to fulfil the requirements of embedded vision systems, as explained in Section 2. There is no hardware architecture that meets perfectly all the necessities. This section gives an overview of available options.

3.1.1 Application-specific integrated circuit (ASIC): ASICs are integrated circuits (ICs) customised for a particular use, rather than intended for general-purpose use. Designers of digital ASICs usually use a hardware description language such as Verilog or VHDL, to describe the functionality of ASICs.

ASICs have the advantages of high performance and low power consumption. They are used only for manufacturing high quantity and long series due to their higher initial engineering cost, so they are not suitable for rapid prototyping. Additionally, they have another important drawback: they are not reconfigurable. This means that once they are manufactured, they cannot be reprogrammed. This lack of flexibility has led to the use of other alternatives such as field-programmable gate arrays (FPGAs). However, some examples of ADAS implementations in ASIC can still be found in the literature [10, 11]. This technology was also used by Mobileye to build its products EyeQ [12] and EyeQ2 [13], which are composed of dual central processing unit (CPU) cores running in parallel with multiple additional dedicated and programmable cores.

3.1.2 Field-programmable gate array: A FPGA is an IC designed to be configured by a customer or a designer after manufacturing. They have lower power consumption and they are better suited for low-level processing than general-purpose hardware, where they clearly outperform them. However, they are not so good for the serial processing necessary in mid and high levels.

3.1.3 Graphics processing unit (GPU): Another hardware architecture especially suited for parallel processing is the GPU. A GPU is a specialised electronic circuit, originally designed to accelerate the creation of images intended for output to a display, which nowadays is also used for general-purpose computing. GPUs have traditionally been considered as power-hungry devices and they are not very frequent yet in in-vehicle applications. However, recent solutions such as the DRIVE PX platform based on the NVIDIA Tegra X1 system on chip (SoC) [14] are very promising.

3.1.4 Digital signal processor (DSP): Traditionally, DSPs have been the first choice in image processing applications. DSPs offer single cycle multiply and accumulation operations, in addition to parallel processing capabilities and integrated memory blocks.

DSPs are very attractive for embedded automotive applications since they offer a good price to performance ratio. However, they require higher cost compared with other options such as FPGAs, and they are not as easy and fast to program as microprocessors.

3.1.5 Microprocessors: Microprocessors are the best option for high-level vision processing. Additionally, they are easy to program, since it is possible to use the same tools and libraries used for standard PC applications. This shortens significantly the learning curve necessary to master a new hardware architecture, which in case of FPGAs and GPUs needs to be specially taken into account.

Advanced RISC Machine (ARM) architectures are leading the microprocessors market, although x86/x64 architectures can also be found. The problem with microprocessors is that they are not very well suited for low-level processing. As a consequence, complex algorithms usually need additional hardware acceleration.

3.1.6 Hybrid or heterogeneous architectures: There is also a growing trend to use SoC that can integrate two or more architectures in the same physical chip. For example, we can have a microprocessor with a FPGA, DSP or GPU, obtaining a more efficient data transfer with less power consumption. Furthermore, SoCs are usually cheaper and have higher reliability than multi-chip solutions.

3.2 Literature review

As shown in the previous overview, there is no ideal platform for implementing a vision-based vehicle detection algorithm. Therefore, this section presents a systematic review of the literature in order to find what the most common embedding options are. The main aim of this systematic review is to address the following research question: what embedded hardware platforms are the most popular among the scientific community to validate vision-based driver assistance algorithms?

Hardware gets obsolete very fast, so it is not worth reviewing papers published a long time ago, since the hardware they used on their work is already outdated. Hence, only papers published since 2012 were considered. As we are only taking into account embedded hardware, papers that only implemented their method in a standard PC were automatically discarded.

Titles, abstracts and full articles were subsequently screened applying the inclusion criteria mentioned. In addition, references of the included articles were checked for other articles eligible for this review (snowball method). As quality criteria, only articles indexed in *Web of Science* and *IEEEXplore* were considered.

The search in these databases using the mentioned inclusion criteria resulted in a total of 33 articles. The selected articles implemented at least one specific driver assistance application in an embedded platform. Articles presenting only generic hardware architectures were discarded. As shown in Table 1, some works implemented more than one driver assistance application. Forward collision warning (FCW), LDW and traffic sign recognition are the most common applications.

The number of papers found each year for each hardware category is detailed in Table 2, where *Mic.* means microprocessor and *HA* means hardware accelerator, which includes FPGAs and other kind of dedicated hardware accelerators excluding GPUs.

It can be noted that most of the works include a microprocessor in their platforms (78.78%). DSPs are included in 30.3% of cases, and FPGAs or other dedicated hardware accelerators in the 33.3%. As Fig. 2 shows, about half of the works (54.54%) use a pure FPGA, DSP or microprocessor solution, probably because from the point of view of the design, it is simpler to use a homogeneous platform. The microprocessor is also the most common option among the works that use a homogeneous architecture, although DSPs are also quite common. In contrast, only in one case is the whole application embedded in a FPGA. The application implemented was a LDW, which is one of the simpler vision-based driver assistance applications from the computational point of view. This is due to the difficulty of implementing the high-level processing part of a computer vision algorithm in a platform that is more suited for SIMD.

Regarding heterogeneous architectures, systems composed of microcontrollers and hardware accelerators are the first choice (27.27% of total cases). Normally, the hardware accelerators are

Table 1 Articles included in the systematic review categorised by application

Embedded driver assistance application	Articles
vehicle detection/FCW	[15–24]
LDW/lane keeping and lane changing	[15, 23–30]
traffic sign recognition	[31–36]
pedestrian detection	[37–39]
night-time vehicle taillight detection	[40–42]
surround view camera	[43, 44]
free space detection	[45]
driver fatigue and distraction monitoring	[46]
full stereo pipeline	[47]

Table 2 Number of publications for embedded vision ADAS per year

Year	FPGA	DSP	Mic.	HA + Mic.	DSP + Mic.	GPU + Mic.	HA + DSP
2012	0	1	2	1	1	0	0
2013	1	0	0	1	1	0	0
2014	0	2	1	3	0	0	0
2015	0	2	9	4	2	1	1
total	1	5	12	9	4	1	1

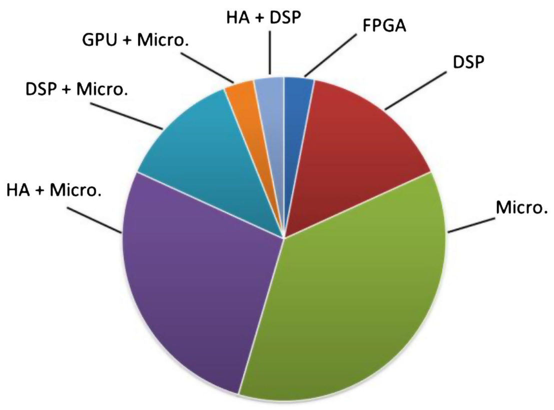


Fig. 2 *Hardware architectures used by the scientific community to embed vision-based ADAS (2012–2015)*

synthesised in FPGAs. SoCs composed of DSPs and microcontrollers are less frequent (12.12% of total cases).

It is worth noting that only one paper was found to implement at least some part of the algorithm in an embedded GPU. On the other hand, there are numerous research works in the literature where part of the vision algorithm is implemented in a GPU using a standard PC. The reason for this difference can be that GPUs have a very attractive architecture to boost vision-based algorithms, and are seen like an interesting platform by a significant part of the computer vision community. However, they have traditionally been very power-hungry devices, and they are not the first choice between experts on embedded systems.

4 Software

Considering that every Original Equipment Manufacturer (OEM) has access to similar-quality hardware and sensors, the embedded software becomes the unique feature OEMs have to add value to their products and differentiate their brand. A massive change in automotive industry is foreseen, where software will play a very important role. Therefore, the software platform used to implement each driver assistance algorithm needs to be carefully considered.

Computer vision applications are implemented in a microprocessor in two main ways: as standalone software, or as a process running on top of an operating system (OS). The first approach obtains better computational results, since it does not have the burden of an OS running on background. However, although the performance decreases when using an application that runs on an OS, it has many other advantages. First, there are great savings in development time and in the maintenance of the system. Second, the non-functional requirements of ADAS software systems are better addressed, which are scalability, extensibility and portability [48]. Third, when using an OS the programmers can focus on the specific computer vision algorithms without having to care about other low-level details. The number of programming errors is reduced when using a higher abstraction level. Last but not least, using a real-time OS (RTOS), the strict reliability and safety requirements of embedded ADAS are better fulfilled.

Ideally, the software for ADAS should be developed for its integration into AUTOSAR environment [49]. Some of the RTOS that are certified for highest ISO 26262 automotive safety integrity level (ASIL) tool qualification level D are: Green Hills Integrity, ElectroBit Tresos Auto-Core OS and Microsar OS SafeContext from Vector [50]. Normally, these options are only used in the industry, since the academia prefers more open options such as Linux-based OS.

5 Design, development, validation and verification

5.1 Introduction: infeasibility of complete testing

The quantification of the dependability of life-critical computer-based systems has been extensively studied in the aeronautics field. In this field, a system is considered ultra-dependable if it has a failure rate of $<10^{-7}$ failures/h [51]. If we translate this to the automotive field, it means that in order to validate the failure rate of a driver assistance system, it is necessary to conduct at least 10^7 h (1141 years) of operational testing. In fact, this number should be much bigger to achieve statistical significance. This leads to the idea of the infeasibility of complete testing [52].

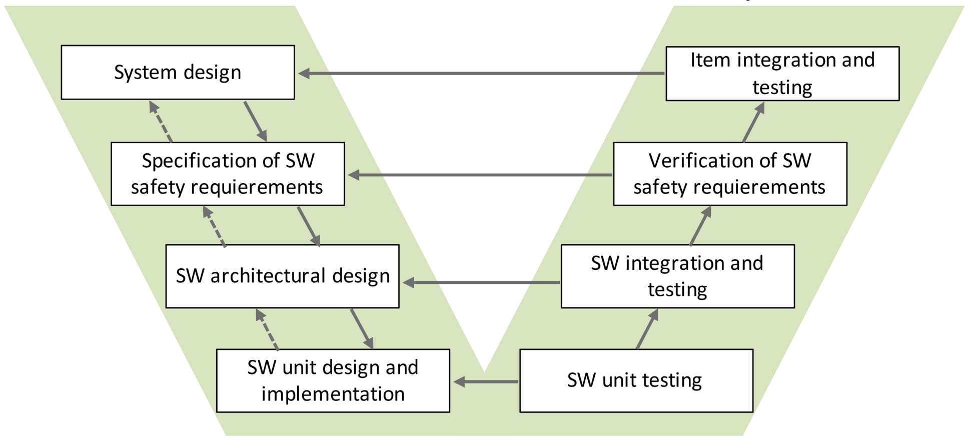
Due to the impossibility of conducting a statistically significant testing to ensure ultra-dependability, other strategies need to be followed. Dependability needs to be considered from the very beginning of the design stage until the last validation test. So design, development and validation concepts cannot be completely decoupled from each other and cannot be implemented in isolated sequential stages. There is a standard that addresses this, ISO 26262, which sets a methodology that guides the hardware and software design, development, validation and verification procedures, following a V-model flow.

5.2 V-model

ISO 26262 is an adaptation of IEC 61508 standard, and it defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems, which includes vision-based driver assistance systems. ISO 26262 is based on a V-model as a reference process model for the different phases of product development.

A simplified version of the V-model for product development at software level is represented in Fig. 3. Starting from the upper-left side, the workflow goes from requirements to implementation. At each step, the system is typically broken down into smaller components. The right side of the V incrementally verifies and validates larger portions of the system. A similar approach is proposed for product development at hardware level.

The V-model described in ISO 26262 is generic enough to be valid for any automotive electronic and electrical safety-related system. Vision-based ADAS contain specific features different from other automotive systems that need to be considered. The



System design
Item integration and testing
Specification of SW safety requierements
Verification of SW safety requierements
SW architectural design
SW integration and testing
SW unit design and implementation
SW unit testing

Fig. 3 *Product development at software level following a V-model flow*

following subsections discuss specific aspects of the design, development, validation and verification steps of vision-based ADAS.

5.3 Design

Not all the operation failures that occur during execution of an ADAS application are responsibility of the algorithm. The hardware and software can also fail due to design errors that do not belong to the vision algorithm itself. Since it is not possible to develop a system with a zero failure rate [53], the ASIL risk level categories are used. A tolerable failure rate is assigned to each application in order to quantify the degree of rigor that should be applied in the development, implementation and verification stages.

There are four ASIL levels, A, B, C and D, arranged in increasing order of integrity requirements. Each hazardous event is assigned an ASIL based on the combination of three parameters: severity (extent of human harm), probability of exposure to operational situations and controllability (ability for persons at risk to take action to avoid harm) [54]. A system that requires an ASIL A would only represent likely potential for minor injuries, while a system that represents likely potential for severely life-threatening or fatal injury would need an ASIL D (see Table 3). A system can be decomposed into several different elements, and then, a high ASIL can be met by having redundant components working together, each one with lower ASIL than the overall system.

This can be achieved implementing, for example, the monitor/actuator architecture proposed in [52]. In this architecture, the primary functions are performed by one module (the actuator), and an additional module is responsible of monitoring the correct functioning of the prior module. If the monitor is designed with an ASIL high enough, the actuator can be designed with a lower ASIL.

5.4 Development

The seamless synchronisation and fusion of the camera output with the rest of vehicle sensor outputs supposes a great challenge. In order to cope with this problem, practitioners use middleware created specifically for automotive applications. Two of the most used ones are EB Assist ADTF [55] and RTMaps [56]. Other options include Polysync, BASELABS and vADASdeveloper by Vector.

Another big challenge of embedding a vision algorithm is to run it in real time in the target embedded hardware platform. The frame rate of the overall driver assistance system (FPS_{ADAS}) is limited by the frame rate of the camera ($\text{FPS}_{\text{camera}}$) and the processing time of the application (T_{proc})

$$\text{FPS}_{\text{ADAS}} = \min\left(\frac{1}{T_{\text{proc}}},\ \text{FPS}_{\text{camera}}\right) \tag{1}$$

Automotive cameras normally have a maximum frame rate of 30 or 60 frames/s. Hence, if the software is able to run in a higher frequency, the camera will be the one imposing the overall frame rate of the system. On the contrary, if the software runs too slow, it will be the bottleneck of the system. It is important to note that the total processing time of the application, T_{proc}, must include the image capture time.

The frame rates and the processing times of the driver assistance systems implemented in the articles studied in the literature review of Section 3 are detailed in Table 4. Some of the articles only indicated the final frame rate, but they did not clarify whether the bottleneck was the camera or the software. For instance, if one only says that the frame rate is 30 frames/s, which is a typical camera frame rate configuration, we cannot know if this is due to the camera limitation or due to the algorithm implementation. Other articles only indicated the processing time of the algorithm, and they did not comment anything about the final frame rate of the overall system. A comprehensive study of a vision-based embedded system performance should include a detailed time profiling of the implemented software and also the overall frame rate of the system considering the selected camera and the rest of the hardware.

We have calculated the median, mean and standard deviation values of the data collected from the literature and the results are listed in Table 5. We have eliminated the outlier of [31], which presented an implementation that was very far from real time. The obtained median frame rate is 27.5 frames/s and the mean value is 25.86 frames/s. These results are in line with what stated in Section 2, where it was said that normally the required real-time frame rate is between 15 and 30 frames/s. The median and mean processing times are 31.99 and 44.89 ms, respectively, which are also in line with the previous statement. In this case, the standard deviation is very high, which is explained by the great differences in the computational complexity of different vision algorithms.

Most of the works in the literature (83.33%) achieve the real-time objective using an image resolution between 320×240 and 752×480 (see Table 4), thus between Quarter Video Graphics Array (QVGA) and Wide Video Graphics Array (WVGA). The most frequent resolutions are 320×240 (20.83%) and 640×480 (33.33%). The decision of choosing a camera image resolution is determined essentially by three factors: computing limitations, the economic cost of the camera sensor and the necessities of the computer vision algorithm. Each vision algorithm demands a minimum image resolution. For instance, traffic sign recognition requires higher resolutions than other algorithms, as it can be concluded from the analysis of the literature (see Table 8 from Appendix).

If it is not possible to run the algorithm in real time with the required resolution, it is necessary to take one of these actions: upgrade the hardware, choose another algorithm or optimise the current one. When the current algorithm is not very far from running it in real time, the latter option is normally chosen. Some good practices and procedures to optimise vision-based driver assistance algorithms are detailed in [18]. Having strict coding rules is also fundamental, such as the ones proposed by the Motor Industry Software Reliability Association (MISRA) [57]. MISRA developed MISRA C and MISRA C++ software coding guidelines, which have the objective to facilitate code safety, portability and reliability in the context of embedded systems.

If the system is hard real time, not only the computational performance is important but also making the performance predictable. Just improving the average performance may not be enough. For instance, pipelining can improve performance on streaming stages but may not help satisfying hard real-time requirements. It is necessary to meet the system deadlines, so the software tasks need to be designed with this in mind. Assigning higher priority to the most time-critical tasks allows using a slower/cheaper processor while satisfying real-time constraints [5].

Vision systems based on supervised learning need to be trained with datasets that cover all the possible cases. In a vision-based driver assistance application it implies capturing data with different illumination, weather, road and driving conditions. Once the video compilation is finished, it is necessary to annotate all the objects and events that have been recorded and need to be detected by the algorithm. Traditionally, this video annotation has been done manually, having dedicated workers labelling each frame. However, as the size of the video databases grow, this solution becomes non-feasible. Video annotation tools, such as Vatic [58] or Viulib's Video Annotator [59], make it easier to build massive, affordable video datasets. The collected video datasets are not only useful for developing new functions and improving current systems, but they are also useful for a first validation in laboratory.

Table 3 Automotive safety integrity levels

ASIL level	Consequences of a failure
A	potential for minor injuries
B	possible major injuries or one fatality
C	possible fatalities
D	possible fatalities in the community

5.5 Validation and verification

Driver assistance or automated driving functions cannot disturb normal driving conditions and should at least be as safe as unassisted manual driving. Therefore, a big effort should be spent on validation and verification stages. According to [60], validation is the assurance that a product, service or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. While verification involves an evaluation of whether or not a product, service or system complies with a regulation, requirement, specification or imposed condition. It is often an internal process.

As explained before, the ISO 26262 standard guides the hardware and software design process and also sets acceptance criteria for each level of integration that must be fulfilled by testing. However, it is a generic approach and it does not detail any testing strategy that can lead to an efficient validation of vision-based driver assistance functionalities. The big problem is that existing analytical methods used in other fields are not useful here, due to the complexity and the wide diversity of driving scenarios and situations. In practice, algorithm verification and validation is currently done using brute-force methods that are extremely costly in money and time. Suppliers spend a lot of time gathering datasets that cover all the possible cases. As a consequence, usually, more time is spent in testing and validating a driver assistance system than developing the algorithm.

Computer vision algorithms have traditionally been tested against video datasets. In vision-based driver assistance algorithms, the same approach is used for a preliminary in-laboratory validation. As explained before, it is extremely costly to gather the required video datasets. Hence, it is now becoming quite common for researchers to release datasets, what facilitates the algorithm testing step for small research groups. Perhaps the best known public dataset is the KITTI Vision Benchmark Suite [61], created by Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, which contains about 180 GB of videos recorded in different conditions. Although open datasets are useful for small research groups and universities, vehicle manufacturers and TIER 1 suppliers have their own datasets that are much bigger, containing several petabytes of video data.

Normally, algorithms are first evaluated on a PC using some performance metrics. The definition of the most common ones are presented in Table 6, where *TP* means number of true positives, *TN* means number of true negatives, *FP* means number of false positives and *FN* means number of false negatives.

The performance results obtained by the articles included in the literature review of Section 3 are presented in Table 7. In some articles, different results were given depending on the driving conditions. In these cases, the average value was considered.

The median, mean and standard deviation performance results are also included in Table 7. The median accuracy, recall and precision values are all above 90%. The standard deviation in the three cases is around 10%, which is reasonable since there are significant differences in the maturity level of the different driver assistance algorithms implemented. The recall is the most used metric as it is calculated in almost all works. Regarding the false positive and false negative rates, the differences here are bigger. The sample size of the false negative rates is too small to draw conclusions, but in the case of the false positive rates, part of the reason of the high standard deviation value might be that some works prioritise maximising the true positive number with the drawback of increasing the false positive number.

It is important to note that it is not possible to make a completely fair comparison as there is not any standard procedure for collecting and labelling the video datasets used for calculating these metrics. Not only this, the length and variety of the datasets differ significantly from one work to another. This leads to a challenge that has not been solved yet: the standardisation of the preparation and validation of evaluation datasets. The average 90% obtained on accuracy, recall and precision metrics also seems quite low for a real deployment. This leads to another unsolved question: what are the minimum threshold values that are required in the performance metrics? This question is related to the concept of infeasibility of complete testing introduced in Section 5.1 and to the commented complexity and wide diversity of driving scenarios and situations.

The testing done in the PC is useful to get some idea of the algorithm's performance, but further testing is needed in the embedded platform. Vision-based embedded systems, as well as other kind of driver assistance systems, are normally tested in three stages [62]:

Table 4 Details about the articles included in the systematic review

Article	Frame rate, frames/s	Processing time, ms	Resolution
[15]	32	31.08	?
[40]	8	132	320 × 240
[31]	0.5	1020	?
[43]	30	?	480 × 204
[41]	12	79.7	720 × 480
[25]	40	?	752 × 320
[26]	22	?	320 × 240
[16]	30	?	320 × 240
[44]	30	?	1280 × 720
[32]	12	?	?
[47]	60	2	752 × 480
[33]	?	16	640 × 480
[17]	18	?	640 × 480
[27]	?	13	?
[28]	30	9.596	320 × 240
[18]	?	11.85	320 × 240
[19]	?	31	?
[20]	30	?	640 × 480
[21]	14	64.29	640 × 480
[29]	16.63	45.31–49.26	640 × 480
[37]	?	65	720 × 480
[42]	5.46	183.16	?
[22] (FWC)	10	48.810	640 × 480
[22] (LDW)	30	32.898	640 × 480
[34]	43	23.25	1920 × 720
[23]	6–8	110	384 × 216
[30]	25	?	?
[38]	30.73	32.54	800 × 480
[39]	20	?	640 × 480
[46]	6	?	320 × 180
[36]	60	?	1920 × 1080

Table 5 Median, mean and standard deviation values for frame rate and processing time

	Median	Mean	Standard deviation
processing time, ms	31.99	44.89	41.84
frame rate, frames/s	27.5	25.86	15.31

Table 6 Performance metrics

Performance metric	Definition
accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$
true positive rate/detection rate/recall	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
false negative rate	$\frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}}$
false positive rate	$\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$

- Testing on simulators.
- Testing on test tracks.
- Testing on public roads.

There are numerous simulators to test driver assistance applications. Among the ones that are free, Racer [63] is probably the most known. Other proprietary options include SiVIC [64] and PreScan [65]. A more detailed review of simulators for testing ADAS can be found in [66].

The next step is testing in test tracks. Validation on the test track is done gradually, first assuring that all components are working independently. Then, driving manoeuvres are tested, starting from the most basic ones, and increasingly adding more complicated manoeuvres to the testing catalogue [62].

The last step is testing on public roads. Since the first DARPA challenges, the importance of extensive on road testing has always been perceived as a core requirement before the public deployment of a driver assistance technology. There are several articles in the literature that present the conclusions obtained after field tests (see [67] for a brief overview). Normally, all tests include instrumented cars with data gathering and recording equipment and extendable in-car platforms.

6 Discussion

The strict requirements and constrains of embedded ADAS guide the decisions taken during the whole cycle of design, development, validation and verification. This paper has shown that there is no ideal implementation platform for vision-based ADAS that meets perfectly all these requirements. The specific features of vision-based algorithms can make one conclude that the best option is a hybrid platform composed of a microprocessor and a hardware accelerator such as a FPGA or a GPU. The low-level processing part would be integrated in hardware, and the rest of the algorithm in the microprocessor. However, only half of the works found in the literature use this option. Almost all the remaining works are based on a pure microprocessor or DSP implementation. This division in the research community could be caused by the difficulty of dealing with hybrid architectures. Even as some hardware/software codesign approaches exist [68], they still involve a much bigger effort than a pure software implementation. A hardware implementation can achieve much better performance with lower power consumption than a software implementation. However, it requires much more time and effort in the design and test, implying higher development costs. Although some reconfigurable options exist, hardware implementations are still significantly less flexible than software options. Having a good balance between hardware and software is essential when designing the whole architecture.

Usually, the first prototype version does not achieve the desired real-time performance and requires some optimisations. However, an optimisation too dependent on the hardware architecture prevents a fast porting to a different hardware or it can even make it impossible. Considering the pace at which hardware's processing speed advances, in the future probably more generous hardware platforms will be used and low-level code optimisation will not be necessary.

It is also important to minimise false positive and false negative rates of ADAS, which are not always caused by the algorithm. Some decisions taken during the embedding stage can also influence. For example, camera calibration, image resolution, infrared illumination in case of driver drowsiness detection and so on. For this reason, it is very important to pay much attention to all these factors, and not concentrate only on the algorithm. It is a frequent error in computer vision practitioners to focus only on the algorithms, without even being aware of the impact of other factors in the final output.

Previous sections have described the current progress in embedding vision-based ADAS. Human drivers are still responsible of monitoring the driving environment in currently

Table 7 Performance results in the articles included in the literature review

Article	Accuracy, %	Recall, %	Precision, %	False negative rate, %	False positive rate, %
[15] (LDW)	?	97.11	?	?	?
[15] (FCW)	?	99.22	?	?	?
[40]	?	79	?	?	10.8
[31]	?	89.9	?	?	?
[43]	?	89.33	?	10.66	14.5
[41]	?	92.08	?	?	?
[25]	97	?	?	?	?
[16]	?	99.29	?	?	0.57
[32]	?	93.24	?	?	?
[33]	?	82	?	?	?
[17]	96	?	?	?	?
[28]	?	95	91	?	?
[18]	?	71.7	75	?	?
[19]	?	98.4	93.3	?	?
[20]	91.3	?	?	?	?
[21]	?	81.78	88.48	?	?
[29]	?	>90	?	?	?
[37]	?	92.14	?	?	?
[42]	?	93.02	?	?	2.81
[34]	98.47	?	?	?	?
[23] obstacle detection	?	98.4	99.4	?	?
[23] lane detection	?	98.1	94.5	?	?
[24] obstacle detection	?	99.04	?	0.96	0.85
[24] lane detection	?	>95	?	?	<2
[38] pedestrian detection	80	36	96	?	?
[38] license plate recognition	74	72	100	?	?
median	93.65	92.58	93.90	—	2.40
mean	89.46	88.26	92.21	—	5.25
standard deviation	10.12	14.39	7.97	—	5.90

deployed ADAS, what corresponds to automation level 1 or level 2 in SAE Standard J3016 [69]. There are still several technical challenges that we need to overcome to advance in embedding more autonomous vision-based driving functionalities (SAE levels 3–5):

- One of the biggest problems when embedding an algorithm is that normally its performance on the target platform is not predictable. Vision algorithms are first programmed in a standard PC, and then are ported to the target embedded platform. The problem is that embedded platforms have significantly less processing power than PCs and they usually have a different architecture, so until you test your algorithm in the embedded platform you are not sure of its final performance. There should be more research focused on performance prediction for image processing algorithms embedded on different platforms. There are some works in the literature focused on this topic [70–73], but there is still much work to be done.
- The current video annotation tools are still mostly manual. It is necessary to advance on more automatic annotation tools in order to reduce the time and cost needed to develop and test new algorithms. Additionally, major improvements in software engineering are needed.
- Outside the autonomous driving sector, safety-of-life critical decisions are never assigned to software systems. For instance, in the medical domain, software is used to support physicians by making recommendations about treatments for patients, but the final decisions are always taken by a human physician. In autonomous driving, the decisions taken can have similar life or death consequences. Major improvements in software engineering are also required here before those decisions can be trusted without a previous human approval.
- Several potential cyberattacks that could affect vision components of future automated vehicles have been identified [74]. Some public demonstrations of hacking attacks with real cars have also been done. Cybersecurity methods to protect against attacks should be further studied.
- In other sectors such as the aerospace industry, the classical approach to ensuring high dependability of systems involves designing in redundancy. It is an effective approach, but it raises a lot the economic cost of products. Cars, in contrast to airplanes, are mass-market products, so the prices need to be kept low. Consequently, new methods should be studied to reduce the dependency on hardware redundancy.
- Current design flows consider safety concerns from the very beginning; however, it is still necessary to incorporate ethical

Table 8 Details about the articles included in the systematic review

Article	Year	Application	Platform	Frames/s	Resolution
[15]	2012	FCW and LDW	DSP: TMS320 DM6437	32	?
[40]	2012	night-time taillight detection	mobile device with 1 GHz CPU and Android 3.0	8	320 × 240
[31]	2012	traffic sign recognition	smartphone: Iphone 3GS	0.5	?
[43]	2012	surround view camera	FPGA + CPU	30	480 × 204
[41]	2012	night-time taillight detection	ARM + DSP: TI OMAP3530	12	720 × 480
[25]	2013	LDW	FPGA	40	752 × 320
[26]	2013	LDW	FPGA with a softcore processor: MicroBlaze	22	320 × 240
[16]	2013	vehicle detection	DSP + Micro.: TIDM3730	30	320 × 240
[44]	2014	surround view camera	2 DSP C66x cores	30	1280 × 720
[32]	2014	traffic sign recognition	FPGA + soft-core processor: Xilinx Virtex 6	12.4	?
[47]	2014	full stereo pipeline	FPGA and mobile CPU	60	752 × 480
[33]	2014	traffic sign recognition	FPGA + PowerPC: Virtex4	?	?
[17]	2014	vehicle detection	PCM-9362 Intel embedded platform	18	640 × 480
[27]	2014	lane keeping and lane changing	DSP: eZdspF2812	?	?
[28]	2015	LDW	ARM + FPGA: Xilinx Zynq	30	320 × 240
[18]	2015	vehicle detection	embedded computer: Nexcom VTC 7120-BK	84	320 × 240
[19]	2015	vehicle detection	mobile device developed by Nokia	?	?
[20]	2015	vehicle detection	mobile devices: Sony Xperia Zl and Gsmart M1V2	30	640 × 480
[21]	2015	vehicle detection	DSP	14	640 × 480
[29]	2015	LDW	Snapdragon 600 embedded processor	16.63	640 × 480
[37]	2015	pedestrian detection	DSP: TI DM648 DSP	?	720 × 480
[42]	2015	taillight detection and traffic sign recognition	CITRIC smart camera platform	5.46	?
[45]	2015	free space detection	FPGA + DSP	?	?
[22]	2015	vehicle detection	multi-core processor, accelerators, and a RISC core: TMPV7506XBG SoC	10	640 × 480
[22]	2015	LDW	multi-core processor, accelerators, and a RISC core: TMPV7506XBG SoC	30	640 × 480
[34]	2015	traffic sign recognition	DSP + ARM: TI DM6467	43	1920 × 720
[23]	2015	FCW + LDW	Tablet: LG Optimus V900 Pad	6–8	384 × 216
[30]	2015	LDW	TDA2x SoC: 2 DSP, 2 ARM cores, 4 HW accelerators	25	?
[24]	2015	LDW	iMX6Q board with an ARM9Q	?	?
[38]	2015	pedestrian detection and license plate recognition	GPU + Micro.: NVIDIA Jetson	30.73	800 × 480
[39]	2015	pedestrian detection	2 mobile devices: Samsung Galaxy Tab Pro T325 tablet and Sony Xperia Z1 smartphone	20	640 × 480
[35]	2015	traffic sign recognition	FPGA + Micro: Xilinx Zynq	?	?
[46]	2015	driver fatigue and distraction monitoring	smartphone: Xiaomi Redmi 1S	6	320 × 180
[36]	2015	traffic sign recognition	FPGA + Mircro.: Spartan-6-FPGA	60	1920 × 1080

considerations. The wide adoption of autonomous or semi-autonomous vehicles promises to dramatically reduce the number of road accidents, but some accidents will be unavoidable and will require the system to choose the best option from the ethical point of view. For example, choosing whether to run over a group of pedestrians or to sacrifice the passenger by driving into a wall [75].

7 Conclusions

This paper provided an insight into vision-based embedded ADAS. The key elements of this field were explained, starting from the features and requirements that these systems have. These features and requirements guide all decisions taken about hardware and software. The balance between the three levels of processing of the implemented vision algorithm (low level, mid-level and high level), is of big importance when designing both the hardware architecture and the finally optimised software. It is also remarkable that it is not possible to fulfil completely all the requirements, so there must be a trade-off between the several design requisites.

This paper also reviews the hardware or software options for embedding vision-based ADAS. To the best of the authors' knowledge, this is the first time that a systematic review about what embedded hardware platforms are the most popular among the scientific community is presented. This study is useful not only to show a comprehensive overview of what hardware architectures are currently used for vision-based ADAS, but also to infer future trends. Regarding software, several important aspects were discussed, such as the design flow, the OSs, the importance of frame rate and image resolution, or the annotation tools that are useful to train the supervised machine learning algorithms. The importance of validation and verification stages is also reflected in the paper, with a brief review of current approaches.

To conclude, it is important to note that there are still some challenges and open research questions that need to be addressed in order to advance towards a more autonomous and reliable driving. The automotive industry and the research community have achieved important milestones but there is still a long way to go, and there are many difficulties to overcome before the final objective of fully automated driving is reached. We believe that computer vision will play a key role here. However, it is not only important to improve current algorithms, but is also essential to focus on developing new methods, tools and architectures to embed them, in order to reduce the burden that this step involves today.

8 Acknowledgments

This work was supported by the EU-H2020 project *inLane*, under the grant agreement number 687458, and by the program ELKARTEK of the Basque Government under the project TIPOTRANS.

9 References

[1] Bengler, K., Dietmayer, K., Färber, B.*, et al.*: 'Three decades of driver assistance systems: review and future perspectives', *IEEE Intell. Transp. Syst. Mag.*, 2014, **6**, (4), pp. 6–22

[2] World Health Organization: 'Global status report on road safety 2015', 2015

[3] Laprie, J.: 'Dependable computing: concepts, limits, challenges'. 25th Int. Symp. on Fault-Tolerant Computing, 1995, pp. 42–54

[4] Aviz, A., Laprie, J., Randell, B.: *'Fundamental concepts of dependability'. LAAS-CNRS, Technical Report N01145*, 2001

[5] Kim, K., Choi, K.: 'SoC architecture for automobile vision system', in Kim, J., Shin, H. (Eds.): '*Algorithm & SoC design for automotive vision systems*' (Springer, Netherlands, 2014), pp. 163–195

[6] Wu, N.: 'High speed CMOS vision chips'. 2011 IEEE 54th Int. Midwest Symp. on Circuits and Systems (MWSCAS), 2011, pp. 1–4

[7] Mukhtar, A., Xia, L., Tang, T.B.: 'Vehicle detection techniques for collision avoidance systems: a review', *IEEE Trans. Intell. Transp. Syst.*, 2015, **16**, (5), pp. 2318–2338

[8] Forster, F.: 'Heterogeneous processors for advanced driver assistance systems', *ATZelektronik Worldw.*, 2014, **9**, (1), pp. 14–18

[9] Buehler, M., Iagnemma, K., Singh, S. (Eds.): '*The DARPA urban challenge: autonomous vehicles in city traffic*' (Springer, 2009)

[10] Darouich, M., Guyetant, S., Lavenier, D.: 'A reconfigurable disparity engine for stereovision in advanced driver assistance systems', *Lect. Notes Comput. Sci.*, 2010, **5992**, pp. 306–317

[11] Mielke, M., Schafer, A., Bruck, R.: 'ASIC implementation of a Gaussian pyramid for use in autonomous mobile robotics'. IEEE 54th Int. Midwest Symp. on Circuits and Systems (MWSCAS), 2011, pp. 1–4

[12] Stein, G.P., Rushinek, E., Hayun, G., *et al.*: 'A computer vision system on a chip: a case study from the automotive domain'. 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05) – Workshops, 2005, pp. 130–130

[13] Mobileye: 'EyeQ2'. Available at http://www.mobileye.com/technology/processing-platforms/eyeq2/, accessed January 2016

[14] NVIDIA: 'NVIDIA DRIVE PX'. Available at http://www.nvidia.com/object/drive-px.html, accessed January 2016

[15] Lin, H.-Y., Chen, L.-Q., Lin, Y.-H.*, et al.*: 'Lane departure and front collision warning using a single camera'. 2012 IEEE Int. Symp. on Intelligent Signal Processing and Communications Systems, 2012, pp. 64–69

[16] Wu, B., Chen, Y., Yeh, C.*, et al.*: 'Reasoning-based framework for driving safety monitoring using driving event recognition', *IEEE Trans. Intell. Transp. Syst.*, 2013, **14**, (3), pp. 1231–1241

[17] Chen, G., Shen, P., Cho, C.*, et al.*: 'A forward collision avoidance system adopting multi-feature vehicle detection'. 2014 IEEE Int. Conf. on Consumer Electronics – Taiwan (ICCE-TW), 2014, pp. 125–126

[18] Nieto, M., Vélez, G., Otaegui, O.*, et al.*: 'Optimising computer vision based ADAS: vehicle detection case study', *IET Intell. Transp. Syst.*, 2016, **10**, (3), pp. 157–164

[19] Wang, X., Tang, J., Niu, J.*, et al.*: 'Vision-based two-step brake detection method for vehicle collision avoidance', *Neurocomputing*, 2016, **173**, pp. 450–461

[20] Jheng, Y., Yen, Y., Sun, T.: 'A symmetry-based forward vehicle detection and collision warning system on android smartphone'. 2015 IEEE Int. Conf. on Consumer Electronics – Taiwan (ICCE-TW), 2015, pp. 212–213

[21] Gu, Q., Yang, J., Zhai, Y.*, et al.*: 'Vision-based multi-scaled vehicle detection and distance relevant mix tracking for driver assistance system', *Opt. Rev.*, 2015, **22**, (2), pp. 197–209

[22] Ozaki, N., Uchiyama, M., Tanabe, Y.*, et al.*: 'Implementation and evaluation of image recognition algorithm for an intelligent vehicle using heterogeneous multi-core SoC'. 2015 20th Asia and South Pacific Design Automation Conf. (ASP-DAC), 2015, pp. 410–415

[23] Petrovai, A., Danescu, R., Nedevschi, S.: 'A stereovision based approach for detecting and tracking lane and forward obstacles on mobile devices'. 2015 IEEE Intelligent Vehicles Symp., 2015, pp. 634–641

[24] Gruyer, D., Livic, I.C., Lusetti, B.*, et al.*: 'PerSEE : a central sensors fusion electronic control unit for the development of perception-based ADAS'. 14th IAPR Int. Conf. on Machine Vision Applications (MVA), 2015, pp. 250–254

[25] An, X., Shang, E., Song, J.*, et al.*: 'Real-time lane departure warning system based on a single FPGA', *EURASIP J. Image Video Process.*, 2013, **2013**, (1), p. 38

[26] Anders, J., Mefenza, M., Bobda, C.*, et al.*: 'A hardware/software prototyping system for driving assistance investigations', *J. Real-Time Image Process.*, 2013

[27] Chiang, H., Chen, Y., Member, S.*, et al.*: 'Embedded driver-assistance system using multiple sensors for safe overtaking maneuver', *IEEE Syst. J.*, 2014, **8**, (3), pp. 681–698

[28] Velez, G., Cortés, A., Nieto, M.*, et al.*: 'A reconfigurable embedded vision system for advanced driver assistance', *J. Real-Time Image Process.*, 2015, **10**, (4), pp. 725–739

[29] Satzoda, R.K., Lee, S., Lu, F.*, et al.*: 'Snap-DAS: a vision-based driver assistance system on a snapdragon TM embedded platform'. IEEE Intelligent Vehicles Symp., 2015, pp. 660–665

[30] Hammond, M., Qu, G., Rawashdeh, O.A.: 'Deploying and scheduling vision based advanced driver assistance systems (ADAS) on heterogeneous multicore embedded platform'. 2015 Ninth Int. Conf. on Frontier of Computer Science and Technology, 2015, pp. 172–177

[31] Koukoumidis, E., Martonosi, M., Peh, L.: 'Leveraging smartphone cameras for collaborative road advisories', *IEEE Trans. Mob. Comput.*, 2012, **11**, (5), pp. 707–723

[32] Giesemann, F., Pay, G., Limmer, M.*, et al.*: 'A comprehensive ASIC/FPGA prototyping environment for exploring embedded processing systems for advanced driver assistance applications'. 2014 Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation, 2014, pp. 314–321

[33] Souani, C., Faiedh, H., Besbes, K.: 'Efficient algorithm for automatic road sign recognition and its hardware implementation', *J. Real-Time Image Process.*, 2014, **9**, (1), pp. 79–93

[34] Yin, S., Ouyang, P., Liu, L.*, et al.*: 'Fast traffic sign recognition with a rotation invariant binary pattern based feature', *Sensors*, 2015, **15**, (1), pp. 2161–2180

[35] Borrmann, J.M., Haxel, F., Viehl, A.*, et al.*: 'Safe and efficient runtime resource management in heterogeneous systems for automated driving'. 2015 IEEE 18th Int. Conf. on Intelligent Transportation Systems, 2015, pp. 353–360

[36] Schwiegelshohn, F., Gierke, L., Michael, H.: 'FPGA based traffic sign detection for automotive camera systems'. 2015 10th Int. Symp. on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC), 2015, pp. 1–6

[37] Chiang, C., Chen, Y., Ke, K.*, et al.*: 'Real-time pedestrian detection technique for embedded driver assistance systems'. 2015 IEEE Int. Conf. on Consumer Electronics (ICCE), 2015, pp. 206–207

[38] Son, S., Baek, Y.: 'Design and implementation of real-time vehicular camera for driver assistance and traffic congestion estimation', *Sensors*, 2015, **15**, (8), pp. 20204–20231

[39] Costea, A.D., Vesa, A.V., Nedevschi, S.: 'Fast pedestrian detection for mobile devices'. 2015 IEEE 18th Int. Conf. on Intelligent Transportation Systems (ITSC), 2015, pp. 2364–2369
[40] Chen, D., Lin, Y., Peng, Y.: 'Nighttime brake-light detection by Nakagami imaging', *IEEE Trans. Intell. Transp. Syst.*, 2012, **13**, (4), pp. 1627–1637
[41] Chen, Y.-L., Chiang, H.-H., Chiang, C.-Y., *et al.*: 'A vision-based driver nighttime assistance and surveillance system based on intelligent image sensing techniques and a heterogamous dual-core embedded system architecture', *Sensors*, 2012, **12**, (12), pp. 2373–2399
[42] Almagambetov, A., Velipasalar, S., Member, S., *et al.*: 'Robust and computationally lightweight autonomous tracking of vehicle taillights and signal detection by embedded smart cameras', *IEEE Trans. Ind. Electron.*, 2015, **62**, (6), pp. 3732–3741
[43] Scharfenberger, C., Chakraborty, S., Färber, G.: 'Robust image processing for an omnidirectional camera-based smart car door', *ACM Trans. Embed. Comput. Syst.*, 2013, **11**, (4), pp. 87:1–87:28
[44] Zhang, B., Appia, V., Pekkucuksen, I., *et al.*: 'A surround view camera solution for embedded systems'. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014, pp. 662–667
[45] Neumann, L., Vanholme, B., Gressmann, M., *et al.*: 'Free space detection: a corner stone of automated driving'. 2015 IEEE 18th Int. Conf. on Intelligent Transportation Systems, 2015, pp. 1280–1285
[46] Manoharan, R., Chandrakala, S.: 'Android OpenCV based effective driver fatigue and distraction monitoring system'. 2015 Int. Conf. on Computing and Communications Technologies (ICCCT'15), 2015, pp. 262–266
[47] Honegger, D., Oleynikova, H., Pollefeys, M.: 'Real-time and low latency embedded computer vision hardware based on a combination of FPGA and mobile CPU'. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2014), 2014, pp. 4930–4935
[48] Ahrens, D., Frey, A., Pfeiffer, A., *et al.*: 'Objective evaluation of software architectures in driver assistance systems', *Comput. Sci. – Res. Dev.*, 2013, **28**, pp. 23–43
[49] AUTOSAR: 'AUTOSAR specifications'. Available at http://www.autosar.org/specifications/, accessed January 2016
[50] Nikolic, Z.: 'Embedded vision in advanced driver assistance systems', in Kisačanin, B., Gelautz, M. (Eds.): '*Advances in embedded computer vision*' (Springer International Publishing, 2014), pp. 45–69
[51] Butler, R.W., Finelli, G.B.: 'The infeasibility of quantifying the reliability of life-critical real-time software', *IEEE Trans. Softw. Eng.*, 1993, **19**, (1), pp. 3–12
[52] Koopman, P., Wagner, M.: 'Challenges in autonomous vehicle testing and validation', *SAE Int. J. Transp. Saf.*, 2016, **4**, (1), pp. 15–24
[53] Stein, F.: 'The challenge of putting vision algorithms into a car'. 2012 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops, 2012, pp. 89–94
[54] Birch, J., Rivett, R., Habli, I., *et al.*: 'Safety cases and their role in ISO 26262 functional safety assessment', in Bitsch, F., Guiochet, J., Kaâniche, M. (Eds.): '*Computer safety, reliability, and security*' (Springer, Berlin Heidelberg, 2013), pp. 154–165
[55] Elektrobit: 'EB Assist ADTF'. Available at https://automotive.elektrobit.com/products/eb-assist/adtf/, accessed January 2016
[56] Intempora: 'RTMaps'. Available at https://intempora.com/products/rtmaps.html, accessed January 2016
[57] MISRA: 'MISRA Homepage'. Available at http://www.misra.org.uk/, accessed January 2016
[58] Vondrick, C., Ramanan, D., Patterson, D.: 'Efficiently scaling up video annotation with crowdsourced marketplaces', in Daniilidis, K., Maragos, P., Paragios, N. (Eds.): '*Computer vision – ECCV 2010*' (Springer, Berlin Heidelberg, 2010), pp. 610–623
[59] Vicomtech-IK4: 'Viulib's video annotator'. Available at http://www.viulib.org, accessed January 2016
[60] 'IEEE Guide – Adoption of the Project Management Institute (PMI) Standard A Guide to the Project Management Body of Knowledge (PMBOK GuiDE) – Fourth Edition', 2011
[61] Geiger, A., Lenz, P., Urtasun, R.: 'Are we ready for autonomous driving? The KITTI vision benchmark suite'. Conf. on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3354–3361
[62] Aeberhard, M., Rauch, S., Bahram, M., *et al.*: 'Lessons learned from automated driving on Germany's highways', *IEEE Intell. Transp. Syst. Mag.*, 2015, **7**, (1), pp. 42–57
[63] Cruden: 'Racer'. Available at http://www.racer.nl, accessed January 2016
[64] Vanholme, B., Gruyer, D., Glaser, S., *et al.*: 'Fast prototyping of a highly autonomous cooperative driving system for public roads'. 2010 IEEE Intelligent Vehicles Symp., 2010, pp. 135–142
[65] TASS: 'PreScan'. Available at https://www.tassinternational.com/prescan, accessed January 2016
[66] Gruyer, D., Choi, S., Boussard, C., *et al.*: 'From virtual to reality, how to prototype, test and evaluate new ADAS: application to automatic car parking'. 2014 IEEE Intelligent Vehicles Symp., 2014, pp. 261–267
[67] Broggi, A., Member, S., Buzzoni, M., *et al.*: 'Extensive tests of autonomous driving technologies', *IEEE Trans. Intell. Transp. Syst.*, 2013, **14**, (3), pp. 1403–1415
[68] Teich, J.: 'Hardware/software codesign : the past, the present, and predicting the future', *Proc. IEEE*, 2012, **100**, (Special Centennial Issue), pp. 1411–1430
[69] SAE J3016_201401: 'Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems', 2014
[70] Henzinger, T.A.: 'Two challenges in embedded systems design: predictability and robustness', *Philos. Trans. R. Soc. Lond. A Math. Phys. Eng. Sci.*, 2008, **366**, (1881), pp. 3727–3736
[71] Saussard, R., Bouzid, B., Vasiliu, M., *et al.*: 'Towards an automatic prediction of image processing algorithms performances on embedded heterogeneous architectures'. 2015 Int. Conf. on Parallel Processing Workshops (ICPPW), 2015, pp. 27–36
[72] Saussard, R., Bouzid, B., Vasiliu, M., *et al.*: 'The embeddability of lane detection algorithms on heterogeneous architectures'. IEEE Int. Conf. on Image Processing (ICIP), 2015, pp. 4694–4697
[73] Saussard, R., Bouzid, B., Vasiliu, M., *et al.*: 'Optimal performance prediction of ADAS algorithms on embedded parallel architectures'. 2015 IEEE 17th Int. Conf. on High Performance Computing and Communications (HPCC), 2015, pp. 213–218
[74] Petit, J., Shladover, S.E.: 'Potential cyberattacks on automated vehicles', *IEEE Trans. Intell. Transp. Syst.*, 2015, **16**, (2), pp. 546–556
[75] 'Autonomous vehicles need experimental ethics: are we ready for utilitarian cars?', arXiv Prepr. arXiv1510.03346, 2015

10 Appendix

See Table 8.