FUNCTIONAL ARCHITECTURE FOR AUTONOMOUS DRIVING AND ITS IMPLEMENTATION

Rihards Novickis
Institute of Electronics and Computer Science
Riga, Latvia
rihards.novickis@edi.lv

Aleksandrs Levinskis
Institute of Electronics and Computer Science
Riga, Latvia
aleksandrs.levinskis@edi.lv

Roberts Kadiķis
Institute of Electronics and Computer Science
Riga, Latvia
roberts.kadikis@edi.lv

Vitalijs Fescenko
Institute of Electronics and Computer Science
Riga, Latvia
vitalijs.fescenko@edi.lv

Kaspars Ozols
Institute of Electronics and Computer Science
Riga, Latvia
kaspars.ozols@edi.lv

***Abstract*— It is predicted that due to such technology-driven trends as electrification, connectivity, autonomous driving and diverse mobility, the automotive market will reach a size of 6.7 trillion USD in 2030. Nevertheless, the composition and architectures of the future autonomous vehicles are still unsettled due to the continuous advancement of sensors and computing hardware, availability and interpretation of new data and safety requirements. Given the existing architectures of the autonomous systems, there is a lack of start-to-end depiction of the self-driving car design and components. This scientific article aspires to serve as a starting guide and boilerplate for the evolution of future architectures of such autonomous systems. The article summarizes the previous research in the field, describes the overall hardware and software composition of the EDI Drive-by-Wire car and investigates the components of the autonomous system: data acquisition, perception algorithms, chosen hardware, software component management, vehicle interfacing, approach to fail-safe and fail-operational functionalities.**

Keywords*— *Autonomous driving, Reference architecture, Drive-by-Wire, Connected and Automated Driving, Self-Driving car, DNNs.

I. INTRODUCTION

The widespread publicity of autonomous driving has drawn a lot of societal attention, led to important political decisions, facilitated development of numerous designated platforms and has consolidated research communities across the world. Nevertheless, the autonomous driving challenge still raises the important issues of continuous hardware evolution, ever-growing interconnection of the devices, continuous algorithm improvement and safety requirements concerning the development, testing and exploitation of the autonomous system.

There is a certain lack of information related to the up-to-date architectures of self-driving cars giving the full sense of the related topics. Research on self-driving cars is still ongoing and level-5 [1] autopilot capabilities, which presume extensive usage in urban environments, are expected within the next 10-15 years [2]. However, this research is mostly related to the intelligence of self-driving cars, their perception systems, processing of sensor data with an emphasis on the application of AI-based algorithms.

In this article we disclose the insights, development methodologies and novel data fusion and processing approaches which were developed at Institute of Electronics and Computer Science (EDI) during the construction and development of the autonomous EDI Drive-by-Wire (DbW) car (Fig 1), which is a successor of the EDI GCDC car [3].

An important criterion for successfully introducing a new technology is societal acceptance. As for autonomous driving, the most important aspects are the fail-operationality and robustness against different damages in the system. Therefore, architecture itself must anticipate imperfection and contain mechanisms for dealing with system faults, thus ensuring fail-operationality. Furthermore, the overall autonomous architecture is subject to the scalability requirement and even the evolution of the requirements themselves.

Fig. 1. EDI Drive-by-Wire car.

The following article examines the previously established architectures and provides detailed insights on proposed architecture. This is followed by conclusions and discussion on the further work.

II. RELATED WORK

In 1985 the public was exposed to the pioneering road-following vehicle "Alvin", which was developed in the famous Autonomous Land Vehicle (ALV) project [4]. Although the autonomous system was based on the "Red minus Blue" segmentation algorithm, it was able to autonomously travel a distance of 4.2 km. The developed system utilized both the vision and ranging sensors and its distributed architecture already incorporated elements of scene modelling, collision avoidance, path planning, vehicle control and data registration.

An impressive work has been done in [5], where a Rapidly Adapting Lateral Position Hadler (Ralph) algorithm was developed. Ralph was probed by measuring the time when the steering wheel's position disagreed significantly with its commanded position. This test was carried out for 4586 km in different weather conditions and 98.1% of the time Ralph was able to steer the vehicle autonomously. Algorithm already supported on the fly refitting for the changing environments.

One of the most influential autonomous driving architectures was developed at MIT for the 2007 DARPA challenge and it was one of the few vehicles to finish the race successfully [6]. The vehicle's system architecture clearly distinguishes between sensor data acquisition, perception and planning & control. The solution used distributed computer architecture based on a UDP mechanism. Furthermore, the solution anticipated the evolution of the system requirements. rom today's perspective the solution was lacking fail-safe functionalities as acknowledged by the authors themselves.

An important turning point in the design of perception-based systems was the 2012 ImageNet Image Classification

competition, which was assuredly won by Krizhevsky, Sutskever and Hinton with their deep-learning-based solution [7]. After this success it became evident that Deep Learning (DL) algorithms bear the state-of-the-art (SoA) potential for a variety of applications [8], including autonomous driving.

The development of autonomous driving systems has been influenced by the publication of ISO26262 standard [9] which covers such important aspects as component development at system, hardware and software levels, testing and safety. Further, the safety aspects of autonomous driving have become a research field in itself, concerning such aspects of the development of autonomous vehicles as architecture, approach to implementation, implications for reconfigurable and adaptive hardware [10].

An influential work has been done in [11], where authors applied engineering design methods to develop a functional architecture which is successfully used by the industry. The work clearly distinguishes between perception, decision & control and vehicle platform component classes, furthermore, authors come to an interesting conclusion of migrating components related to trajectory execution to the vehicle platform itself, thus suggesting its association with vehicle manufacturers. The proposed approach is aware of ISO26262 standardization procedures and anticipates parallel work of multiple teams with distinct development tools.

The previous progress has led to a research which is more devoted to meeting the industrial needs [12], e.g. by concentrating on the operation aspects, by developing real-time processing models and exploring the capabilities of specialized multicore hardware.

Although the research field of autonomous systems is very active and covers a wide range of topics and technical aspects, there is a certain lack of a start-to-end depiction of the self-driving car design and its components. This article aspires to fill this hole and facilitate the further improvement of the autonomous systems.

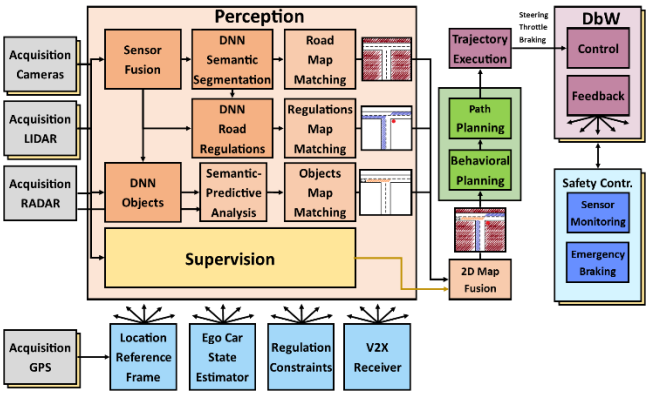

Fig. 2. Functional architecture of the EDI DbW car.

III. Approach

The functional architecture of the EDI DbW car shown in Fig 2 is broken down into the following high-level component groups: perception (sensors, data acquisition, data fusion, environment interpretation, perception fusion), path and behavioral planning (path planning, trajectory update, and execution, actuator signal generation), global state estimation (ego car's state, traffic regulations, V2X interpretation, vehicle platform) and safety functionalities (emergency braking, sensor state estimation). Further we describe the motivation behind these components, their functionality and realization.

A. Sensing hardware

To safely deploy an autonomous car in a real-world environment, it must be able to sense and perceive the surrounding environment with high accuracy and noise resistance as well as precisely understand its position, communicate with other vehicles and infrastructure as well as perform in real-time.

It has been shown that these perception requirements are achievable by adopting sensor data fusion and sensor redundancy [13], [14], [15]. Therefore, to increase accuracy and noise resistance as well as enable fail-safe and fail-operational functionality, multiple sensor modalities (cameras, RADARs and LiDAR) are combined, providing the necessary coverage of the environment (see Fig. 3).

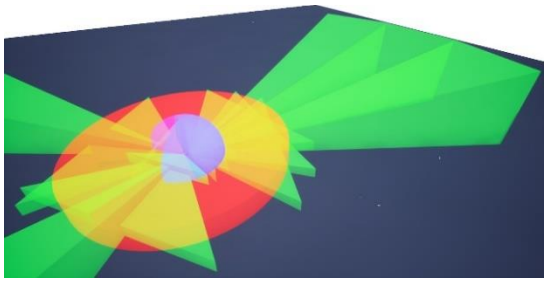

Fig. 3. Field of View of the EDI car using cameras, RADARs and LiDAR (Modeled using Unreal Engine 4).

For location reference frames (combines data from Ego car estimator and GPS to create a coherent coordinate system, which is used by many other components) a GPS RTK + IMU unit is used, while for communication with other vehicles and infrastructure (e.g. road signs, traffic lights) a Vehicle to Everything (V2X) system is used.

B. Component Communication Architecture

During the research and development process, the application may have to go through a multitude of modifications – change or swap of components, modification of component configuration parameters, a need to instantiate the same component multiple times with different configurations, change of data paths, etc. Furthermore, such popular frameworks like ROS [16] introduce overhead [17], which hinders their applicability in such real-time control system as an autonomous vehicle.

The proposed component management solution consists of two frameworks - ***compage*** [1] and ***icom*** [2] - their principal operation is illustrated in Figure 4. *compage* enables a convenient way of managing, configuring and instantiating software components, while *icom* ensures such communication paradigms as request-reply, publish-subscribe and push-pull.

The modularity of the *compage* framework is achieved by utilizing a particular feature of the executable and linkable file format - custom sections - which permits the creation of component-specific data structures within any source file. The *icom* framework is a wrapper around ZMQ messaging library [18] with a zero data copy feature, which can be used to preserve the communication bandwidth and is especially

[1] https://gitlab.com/rihards.novickis/compage.git

[2] https://gitlab.com/rihards.novickis/icom.git

important for handling imaging data. For more information on the frameworks refer to the provided repositories.

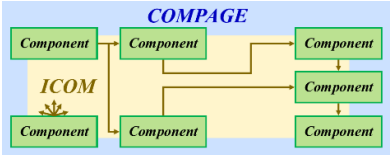

Fig. 4. Visual representation of software component management and intercommunication frameworks.

C. Perception Software

Modern DNN-based perception systems still lack in robustness, indicated by the research on adversarial attacks on classification [19], object detection [20], and semantic segmentation [21-22] models. To develop a robust perception system, we propose a multimodal approach that combines several DL-based models with a model trained to supervise the reliability of system's components.

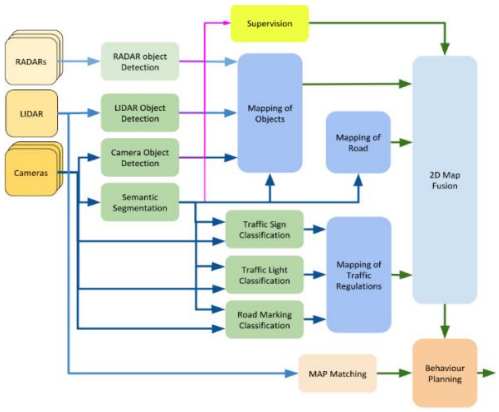

Fig. 5. Perception software overview.

The software of the proposed perception system consists of several ANN-based and map making modules depicted in Figure 5. The system gets data from sensors (RADAR, LiDAR, camera) and outputs a bird's-eye view of vehicle surroundings (a 2D map). The shown architecture is the variation that includes redundancies for increased reliability. Smaller variations with fewer modules are also possible.

1) Deep Neural Networks

The sensor data are first processed by several deep neural networks.

Camera object detection module receives data from cameras. If the embedded computation power is sufficient, data from each camera is processed by its own object detector, but in the case of fewer resources, the camera frames of different cameras are combined into a single image and run through a single model. This model is a deep convolutional neural network, and our current choice is YOLOv3 architecture [23]. It is chosen because of the SoA detection results coupled with the fast inference. Furthermore, when confronted with hardware limitations, the size of the YOLO model can be changed without retraining the model.

RADAR object detection could also be implemented as a deep neural network model; however, in this system we use the built-in capabilities of Continental RADARs which provide detection of moving points and clusters.

Semantic Segmentation DNN processes the same camera frames as the object detector, but it attaches a class label to every pixel of the frame. This is crucial to determine where the vehicle can move as well as segments of road signs and markings to be cut out for smaller CNN-based classifiers. A MobileNet-based DeepLab model has been chosen for the segmentation task [24]. The DeepLabv3 achieves one of the best results on Cityscapes benchmark [25], while the MobileNet [26] backbone provides a computationally efficient feature extraction. In [21], DeepLab v2 model also proved to be more robust against different adversarial attacks when compared to other accurate segmentation methods such as PSPNet [27], CRF-RNN [28], DilatedNet [29], SegNet [30], E-Net [31], ICNet [32], and FCN [33]. Authors of [21] suggest that networks that perform multi-scale processing (e.g. DeepLab v2) are preferred in safety-critical applications.

Road Regulation DNN modules are conventional CNNs (LeNet [34] and similar sized CNNs) that classify paint markings, traffic regulation signs, and traffic lights. The Semantic Segmentation module indicates which frame region should be processed by one of the three classifiers.

2) Mapping

The information gained by perception ANNs is used to create three different maps of the vehicle's surroundings.

Mapping of Objects module receives object information from RADAR Object Detection, LiDAR Object Detection, Camera Object Detection, and Semantic Segmentation modules. It combines the information and maps it onto a 2D map. The areas furthest from the vehicle use only RADAR information, while objects in closer regions are detected by several modalities. At this stage, all the detected objects are put on the map, even if the same object is detected several times by different sensors. The detection method and a confidence score are also stored for each of the objects.

Mapping of Road module receives segmented images from the Semantic Segmentation and uses them to create and update a 2D map of road and non-drivable surroundings.

Mapping Traffic of Regulations module uses information from three classification networks to create a 3D map of labeled road signs, traffic markings on the road, and current signals and positions of traffic lights.

2D Map Fusion module combines the maps from the three previous modules and creates a final map in which drivable regions are depicted based on the extent of the road, detected obstacles, and traffic regulations. All the inputs of the fused map must create a coherent image. Further, multiple same object detections can be used to inject redundancy and make the system fail-aware and even fail-operational. We propose another module that would give information to the map-fuser about the current reliability of each of the modalities.

Supervision module is an ANN that receives outputs of all object detectors as well as an image segmentation module. The outputs of the supervisor indicate the trustworthiness of the detectors. While each separate detector and segmentator can be trained separately, using different training datasets, the supervisor needs to be trained with complete sensor data combination. During the training, data from different sensors and detection modules is intentionally corrupted.

The corresponding training label indicates which input should be detected as corrupted, e.g. when Camera Object Detection module output is intentionally corrupted, the supervisor should learn to indicate the untrustworthiness of that module. When data from cameras is corrupted, the

untrustworthiness should be ascribed to all camera-based modules (Camera Object Detection, Semantic Segmentation, Traffic Sign Classification, Traffic Light Classification, Road Marking Classification). The trained supervisor helps the 2D Map Fusion module to create the most believable map. It also affects the behaviour planning and allows the autonomous vehicle to detect faults in the sensors or processing modules. If a non-critical module is detected as faulty (for example a LiDAR has become temporarily untrustworthy, but the cameras and RADARs are still resulting in coherent and reliable perception), the vehicle detects fault, yet it is still fail-operational. In such a case, the vehicle can continue its course but warns the owner of increased risks and needed repairs. However, when a critical failure in perception is detected, for example, the system does not reliably segment the drivable road, the vehicle will perform an emergency stopping or demand a human driver to take over the control of the vehicle.

D. Behavioral Planning and Path Planning

Behavioral Planning component determines the behavior of the vehicle, e.g. where it is necessary to stop or slow down according to traffic regulations or due to some other traffic member, the Path Planning component uses this information and creates feasible trajectory. The concept of the path planning component is inspired by [6] with the idea of reducing the planning challenge to two spatial dimensions, i.e. bird's eye view.

E. Embedded Systems

Although the autonomous vehicle architecture anticipates improvement of new hardware, EDI DbW car distinguishes between High-Performance and Safety controllers, which are implemented using hardware available at the time.

High-Performance Controller. The computational system of EDI DbW car is based on the extendable NVIDIA PX2 embedded system, which is used as the high-performance controller to execute algorithms and to ensure fail-aware, fail-safe and fail-operational functionality. If one of the high-performance platforms fails, the decision making module of the safety controller utilizes available vehicle state information to prioritize the feasible control signals.

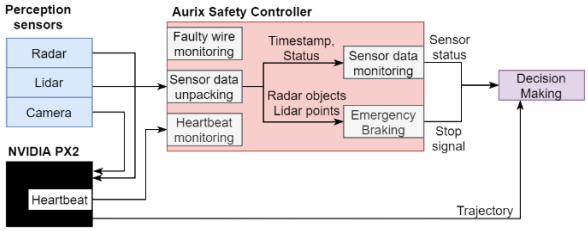

Fig. 6. Safety controller's software architecture.

Safety Controller. In order to achieve fail-safe and fail-operational functioning of the whole system, we implement a specialized safety controller. On hardware level, safety is achieved by using Aurix TC297 Tricore lockstep processor that performs redundant computations and provides functionality designed for safety critical applications. On software level, safety is achieved by sensor and high-performance controller health monitoring as well as emergency braking (see Figure 6).

Health monitoring is performed by checking physical connections to sensors as well as received data. Data are unpacked and such information as sensor's status and timestamps are used to determine sensor's health. Additionally, a heartbeat mechanism is used to monitor the state of the high-performance controller.

Safety controller also performs emergency braking in case of unavoidable collision using LiDAR or RADAR data depending on sensor health. Safety controller's firmware is based on Arctic Core - Arccore's open-source solution and industry-driven AUTOSAR standard. It contains OSEK/VDX grade operating system and hardware independent drivers.

F. Variable Data Acquisition (VDA)

To better utilize system resources (e.g. data bandwidth, computational power) the design of variable data acquisition (VDA) unit was proposed. This unit's main responsibility is to interact with the behavioral planning unit and redirect data flow from the sensors to the processing unit, depending on the driving scenario. It also is in charge of sampling rate adjustment for the sensor.

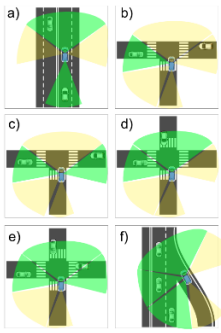

Fig. 7. VDA operation within different AD scenarios.

The VDA concept is beneficial for use cases with constrained bandwidth, e.g. RADAR subsystem and CAN bus. The developed system utilizes five RADARs and CAN bandwidth is not sufficient to run all of them at the highest rates. However, it is possible to change the sampling rate of the object detection for each RADAR depending on the AD (Autonomous Driving) scenario. The most frequent scenarios are the following (Fig. 7):

a) Performing AD on highways, front and rear RADARs are sampled more frequently.

b) Performing AD right turn on T-shaped intersection, left RADAR is sampled more frequently.

c) Performing AD left turn on T-shaped intersection, left and right RADARs are sampled more frequently.

d) Performing AD right turn on X-shaped intersection, left and front RADARs are sampled more frequently.

e) Performing AD left turn on X-shaped intersection, left, right and front RADARs are sampled more frequently.

f) Performing AD on highway ramp (entering highway), sideway, left and front RADAR are sampled more frequently.

G. Drive-by-Wire System

One of the crucial components used in autonomous vehicles is the DbW system, which essentially enables control of the vehicle actuators. Such systems were first introduced in the aircraft industry to enable advanced assistance for pilots by introducing so-called fly-by-wire functionality [35]. The modern cars are equipped and shipped with DbW components due to the OEM investment into ADAS systems and such functionalities as Forward Collision Warning (FCW) and

Lane Keeping Assist System (LKAS). Such functionally mostly implies Level 3 Autonomous Driving (AD) according to the SAE levels of automation [1].

Despite the DbW availability in modern cars, there is an absence of available technical documentation. Luckily, there are existing third-party solutions that enable these essential capabilities by introducing an intermediate system. However, such systems are mostly relying on a car model, are quite expensive and can even exceed the price of the entire vehicle [36-37]. Therefore, an open source solution has been chosen, which is based on the Open Source Car Control (OSCC) project [38] and is a reference design for Kia Soul EV.

Fig. 8. Casing of the adapted DbW unit.

Essentially OSCC utilizes Electrical Power Assisted Steering of the KIA SOUL EV to enable DbW solution without deploying any assisting mechanical solutions. The main idea of the OSCC is the implementation of "signal-spoofing" technique for analog signals between a desired sensor and ECU. The advantage of the "signal-spoofing" technique is its universality, it doesn't not require having any information about ECU CAN interface, other than sensor analog signal range, which can be measured experimentally.

The original reference design of the OSCC system has been modified with an addition of a SBC (Single Board Computer) to enable Ethernet connectivity complimenting the existing CAN communications. This permits the deployment of higher-level services on the SBC. The DbW unit has been tested running Car-in-the-Loop setup which is further described in section IV of this article.

IV. Vehicle Setup and Approach to Algorithm Validation

Even with an established functional architecture of the autonomous vehicle, the actual setup of the car can be quite a challenging task and one must take into consideration a considerable effort required to set up the hardware and software components of such an involved real-time system To perceive the world and surrounding objects, we use several sensors (sensor systems) - twelve SEKONIX SF3323 Cameras, five Continental ARS-408 RADARs, HDL-32 LiDAR, GPS RTK + IMU (for location reference frame, ego car state estimation and regulation constraints) and EDI V2X communication device (developed in H2020 ECSEL Autodrive project [39]).

TABLE I. Installed sensors and their specifications.

Sensor	FoV	Resolution / Range	Rate
Camera	50 -120 deg	1920 by 1200 pixels	30 Hz
RADAR	3-60 deg (near, mid, far)	1 - 200 m	10 - 100 Hz
LiDAR	50-70 deg	1 - 70 m	5 - 50 Hz

The fail-aware, fail-safe, fail-operational EDI V2X device is using 3 different protocols: IEEE 802.11p, LTE/4G cellular data network and LoRA, while operating in 3 different frequencies: 5.9 GHz, 1.8, 2.6 GHz, 863-870 MHz.

The autonomous driving computing platforms are rapidly evolving as more and more corporations are recognizing the growing market [40]. For the implementation of the High Performance Controller we have chosen a specialized NVIDIA PX2 platform, whose main distinctions are two discrete computing systems each consisting of an SoC and a discrete GPU, high-speed Ethernet switch and capability of simultaneously interfacing 12 GMSL cameras. The Safety Controller has been implemented using Infineon's Aurix TC297 Tricore lockstep processor and both controllers are interfaced using Ethernet and CAN interfaces.

High Performance Controller is running Linux and to facilitate the potential upgrade/change of the computing hardware while not neglecting the real-time requirements, we adopt black board software development pattern by utilizing our *compage* and *icom* frameworks. Each component is responsible for a localized task, such as specific sensor data acquisition, inference of DL algorithm or planning. This enables agile development by different technical teams and facilitates experimentation with algorithms and hardware.

Testing and initial validation of the system components is being done utilizing Hardware-in-the-loop (HiL) setup with a simulated environment provided by an open-source CARLA simulator [41]. The CARLA project is an active project, which is continuously maturing and provides such important features as simulated sensors and ground truth segmentation of the camera data. Furthermore, HiL tests are extended to a Car-in-the-Loop (CiL) setup, so that the generated control signals are sent to and back from the actual car. The only distinction from the HiL setup is that the entire car is being used for the control signal validation which are generated by PID or MPC models and are connected to the DbW system. This gives the possibility of testing the entire data processing path from the sensor data acquisition, to the control of a real vehicle.

V. Conclusions

We have presented a functional architecture of an autonomous vehicle and proposed a series of choices and solutions to implement it by using the computing and sensing hardware available at the time. The article outlines the main elements of the autonomous vehicle and its development ranging from a proposed approach to the software component management and perception algorithm pipeline, to the aspects of fail-safety and reference hardware guidelines.

Furthermore, we have extended this reference architecture and have proposed a novel concept of sensor data bandwidth management in a form of VDA feature to focus sensor data acquisition according to the AD scenario. Moreover, we have described the application and adaptation of OSCC DbW system and have shared our component testing workflow, including the validation in Car-in-the-Loop setup.

A hybrid architecture consisting of High Performance and Safety controllers has been proposed with the appropriate hardware choice for the implementation of such a system. This article aspires to serve as a starting guide and boilerplate for the evolution of future architectures of autonomous systems. EDI DbW car will further be used to develop, test and validate AI-based perception algorithms in relation to the fail-operational and fail-safe performance of the car.

VI. Acknowledgments

This work is the result of activities within the "Programmable Systems for Intelligence in Automobiles" (PRYSTINE) project, which has received funding from ECSEL Joint Undertaking under grant agreement No. 783190 and from specific national programs and/or funding authorities.

References

[1] SAE International, "Surface Vehicle Recommended Practice: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles", 2018.

[2] European Commission, EU Road Safety Policy Framework 2021-2030 - Next steps towards "Vision Zero", Brussels, June, 2019.

[3] G. Strazdins et al., "Team 'Latvia' GCDC 2011 Technical Paper," p. 9.

[4] M. A. Turk, D. G. Morgenthaler, K. D. Gremban, and M. Marra, "VITS-a vision system for autonomous land vehicle navigation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 3, pp. 342–361, May 1988, doi: 10.1109/34.3899.

[5] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," IEEE Expert, vol. 11, no. 2, pp. 19–27, Apr. 1996, doi: 10.1109/64.491277.

[6] J. Leonard et al., "A Perception-Driven Autonomous Urban Vehicle," in The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, M. Buehler, K. Iagnemma, and S. Singh, Eds. Berlin, Heidelberg: Springer, 2009, pp. 163–230.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[8] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11–26, Apr. 2017, doi: 10.1016/j.neucom.2016.12.038.

[9] 14:00-17:00, "ISO 26262-1:2011," ISO. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/34/43464.html.

[10] M. Carre, E. Exposito, and J. Ibanez-Guzman, "Challenges for the Self-Safety in Autonomous Vehicles," in 2018 13th Annual Conference on System of Systems Engineering (SoSE), Paris, Jun. 2018, pp. 181–188, doi: 10.1109/SYSOSE.2018.8428718.

[11] S. Behere and M. Törngren, "A functional reference architecture for autonomous driving," Information and Software Technology, vol. 73, pp. 136–150, May 2016, doi: 10.1016/j.infsof.2015.12.008.

[12] Mezzetti, E., et al. AURIX TC277 Multicore Contention Model Integration for Automotive Applications. 2019, pp. 1202–03, doi:10.23919/DATE.2019.8715202. Scopus.

[13] Xu Danfei, Dragomir Anguelov, and Ashesh Jain. "Pointfusion: Deep sensor fusion for 3d bounding box estimation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[14] Cho, Hyunggi, et al. "A multi-sensor fusion system for moving object detection and tracking in urban driving environments." 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014.

[15] Chadwick, Simon, Will Maddetn, and Paul Newman. "Distant vehicle detection using radar and vision." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

[16] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler and Andrew Ng "ROS: an open-source Robot Operating System", Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, Kobe, Japan, 2009.

[17] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in 2016 International Conference on Embedded Software (EMSOFT), Oct. 2016, pp. 1–10, doi: 10.1145/2968478.2968502.

[18] ØMQ - The Guide - ØMQ - The Guide. http://zguide.zeromq.org/page:all.

[19] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[20] Chen, S. T., Cornelius, C., Martin, J., & Chau, D. H. P. (2018, September). Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 52-68). Springer, Cham.

[21] Arnab, A., Miksik, O., & Torr, P. H. (2018). On the robustness of semantic segmentation models to adversarial attacks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 888-897).

[22] Shen, G., Mao, C., Yang, J., & Ray, B. (2019). Unrestricted Adversarial Attacks for Semantic Segmentation. arXiv preprint arXiv:1910.02354.

[23] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[24] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.

[25] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3213-3223).

[26] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[27] Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2881-2890).

[28] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., ... & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In Proceedings of the IEEE international conference on computer vision (pp. 1529-1537).

[29] Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122.

[30] Badrinarayanan, V., Handa, A., & Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293.

[31] Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147.

[32] Zhao, H., Qi, X., Shen, X., Shi, J., & Jia, J. (2018). Icnet for real-time semantic segmentation on high-resolution images. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 405-420).

[33] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

[34] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[35] I. Nicolin and B. A. Nicolin, "The fly-by-wire system," INCAS BULLETIN, vol. 11, no. 4, pp. 217–222, Dec. 2019, doi: 10.13111/2066-8201.2019.11.4.19.

[36] https://www.abdynamics.com/en/products/track-testing/drive-by-wire/flex-0.

[37] https://www.dataspeedinc.com/adas-by-wire-system/.

[38] https://github.com/PolySync/oscc.

[39] H2020 ECSEL "Advancing fail-aware, fail-safe, and fail-operational electronic components, systems, and architectures for fully automated driving to make future mobility safer, affordable, and end-user acceptable" (Autodrive) project, GA No. 737469. https://autodrive-project.eu/.

[40] "Semi-Autonomous and Autonomous Vehicles Market by Technology, Components, Region - 2021 | MarketsandMarkets." https://www.marketsandmarkets.com/Market-Reports/near-autonomous-passenger-car-market-1220.html.

[41] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, "CARLA: An Open Urban Driving Simulator," Proceedings of the 1st Annual Conference on Robot Learning, p. 1-16, 2017