The role of cloud-computing in the development and application of ADAS

Cristian Olariu
Innovation Exchange
IBM Ireland
Dublin, Ireland
cristian.olariu@ie.ibm.com

Juan Diego Ortega
Intelligent Transport Systems and Engineering
Vicomtech
San Sebastian, Spain
jdortega@vicomtech.es

J. Javier Yebes
Intelligent Transport Systems and Engineering
Vicomtech
San Sebastian, Spain
jyebes@vicomtech.es

***Abstract*—This work elaborates on the cycles involved in developing ADAS applications which involve resources not permanently available in the vehicles. For example, data is collected from LIDAR, video cameras, precise localization, and user interaction with the ADAS features. These data are consumed by machine learning algorithms hosted locally or in the cloud. This paper investigates the requirements involved in processing camera streams on the fly in the vehicle and the possibility of off-loading the processing load onto the cloud in order to reduce the cost of the in-vehicle hardware. We highlight some representative computer vision applications and assess numerically in what network conditions offload to cloud is feasible.**

Keywords—ADAS, Cloud-Computing, Cloud Offload, Computer Vision, GPU processing, Image processing, Deep Learning, Driver Assistance, Autonomous vehicles

I. Introduction

Self-driving car technology is becoming a reality and the next steps in its availability to consumers involve the reduction of cost of in-vehicle processing hardware and legislative changes to accommodate scenarios where a human is not behind the wheel. This paper focuses on the first problem, wherein Advanced Driver Assistance Systems (ADAS) processing hardware required to run environment detection is still too expensive for mainstream adoption. The usage of Graphical Processing Units (GPUs) for computer vision has drastically improved the execution time [1] however these are costly, need to be automotive grade, and are energy hungry which also drives the fuel consumption up.

Computer vision is a research field which attracted a lot of attention in the recent years mainly due to its applications. The pioneering applications were handwriting recognition where the focus was on accuracy and has gradually transitioned to object detection and more advanced pattern recognition that mimics the human brains' activity while performing certain tasks. Medicine is another example where computer vision has become an assistant to doctors in analysing x-ray imagery. The most demanding applications are those in which the output of the analysis has a hard time deadline – an object detected too late can lead to self-driving cars' inability to avoid object and thus cause accidents. One of the most challenging task is to develop computer vision capable to drive both in highway scenarios, where the environment is controlled by unidirectional movement and clear lane markings, and an urban scenario where object movement is omnidirectional and objects can vary from vehicles to road furniture and buildings and more importantly, persons.

Matching patterns directly with the input signal has proven unfeasible for real-time applications [2] and implementations using neural network have achieved satisfactory processing times [2] The implementation of neural networks can take advantage of the GPU's internal architecture, cutting the execution time by an order of magnitude [2] compared to Central Processing Units(CPUs). However due to cost constraints explained earlier in the automotive scope, one question arises, namely, whether or not some of the computational efforts can be offloaded to external systems? The only viable solution to move the data from the vehicle to other systems for analysis is over a wireless technology capable to support the required transfer speeds. It is only technology such as 5G that promises to deliver the architecture and transfer speeds capable of supporting such offloading function.

This paper looks into some of the computer vision applications that run in a vehicle, giving concrete numerical examples of processing requirements. A numerical analysis is then performed to investigate on the network requirements and feasibility of offloading data to the cloud. This is done in the context of the research and exploitation cycles involved in the development of ADAS applications built on top of computer vision.

The rest of the paper is organized as follows. Section II highlights relevant works in the field, Section III gives examples of ADAS applications that rely on video input to obtain information about the environment and the driver, Section IV gives a numerical analysis into the feasibility of offloading some of the computer vision applications onto the cloud, and Section V concludes the paper.

II. Literature Review

In [3] the authors investigate the effect of network delay onto the deviation from course of car controlled from the cloud. The work also proposes and architecture that employs Mobile Edge Computing (MEC) as a means of reducing the latency between the control entity (e.g. EUTRAN Node B) and the vehicle. One of the important takeaway message from the work in [3] is that a network delay over 150ms renders any remote control as having a 40% deviation from trajectory, which actually means that the network constraints should be much lower than that value.

ISBN 978-90-827970-1-5 © EURASIP 2018

In [4], the authors investigate the ADAS applications that can be offloaded from the vehicle to MEC nodes which in their work are located in the Road Side Units (RSUs), while the network configuration is done using Software Define Networking (SDN) concepts. Their work proves that the network and compute architecture can be designed to cope with the data volumes and latency requirements when the 5G architectural features are employed. Similar conclusion is also reached by other works, such as [5].

In [6] the authors highlight the capabilities of the 5G network for mobility scenarios, where for a 28GHz 5G spectrum allocation a vehicle travelling at 100 km/h can achieve a stable connection of 1.2Gb/s which is the value we use in our numerical analysis.

The general goal of computer vision is to extract some interpretable and high-level information from matrices of numbers, which are the images and sequences of them. The current state-of-the-art has moved from classical approaches [7] to Deep Learning pipelines [8] that have overwhelmingly outperformed the first ones in many tasks, like ADAS-related [2] which are treated in this work. This has been also possible due to the existence of large annotated datasets and powerful GPU computing hardware. However, allocating the required systems for their deployment in vehicles imposes several constraints and dedicated resources depending on the application. These requirements and their relation to Deep Learning trend for Real-Time applications in Automated Vehicles are reviewed in [9], which also proposes a FPGA implementation as an efficient alternative with some restrictions and future trends.

III. ADAS Data collection and Processing Requirements

ADAS is a field of research and strong industrial development within the area of Intelligent Transportation Systems (ITS). Mainly due to safety concerns towards reducing the number of fatalities on the roads worldwide [1]. ADAS can be also seen as a reduced and individually tackled set of challenges within the scope of driverless vehicles [2]. Despite the early start of exploratory work in autonomous vehicles in 1986, the area has recently attracted a lot of interest from academy and large amounts of investment from industry [2]. Most of the systems can still be considered as separate dedicated modules for specific ADAS functions with increased complexity along the years. Nevertheless, there exist an important force integrating several sensors in the car and working on the fusion of their data streams. This imposes several requirements both on the acquisition and processing of the data to obtain meaningful patterns and semantics which are helpful for the ADAS and driverless tasks.

Furthermore, recent advances on big data processing based on Deep Learning approaches have pushed forward the current state of the art. Deep Learning has proved to be a powerful mechanism to learn complex data patterns yielding high detection rates compared to traditional computer vision algorithms. However, this has required the use of large datasets and powerful hardware devices based on latest advancements in GPU technology [8].

A. ADAS Computer Vision Applications

The following sections describe a set of representative ADAS applications within the state of the art. They are focused on camera and LiDAR sensors which are the ones with highest bandwidth and computation requirements. They rely on 2D and 3D modalities to perceive the environment surrounding the vehicle and to monitor driver behaviour on board.

Multi-lane detection

Lane detection and modelling methods detect the presence of lane markings in the scene and fit a road model to them to determine the position of the ego-vehicle within its lane, the presence and curvature of adjacent lanes, and the type of lane markings. Historically, lane detection was the first effective computer vision method used in the automotive industry for ADAS. As a result, many approaches exist which provide accurate results and low computational cost. Implementations on standard ARM processors @1GHz can run in 10ms [10], while multi-lane road segmentation approaches can take from 6 to 1200ms on GPU-enabled HW [11].

Object detection

This is the function which determines the position, size and/or other spatial properties of specific types of elements in an image. Typically, dictionaries for specific scenarios determine which classes are detectable. In the context of ADAS, these classes include elements such as car, pedestrian, bicycle, truck, pole, traffic sign, etc. It is crucial to provide a reliable detection as the car is sharing the road with many traffic participants, particularly in urban areas. This task is challenging because of the wide variety of object appearances and occlusions caused by different objects or infrastructure in the observed scene. Besides, there are illumination changes and other artefacts depending on sensor modality. A comprehensive review of object detection methods is provided in [2]. On the one hand, 2D object detection is carried out on images from cameras to find the location of bounding boxes around the objects of interest. On the other hand, 3D object detection is performed on the fusion of 3D point clouds from LiDAR and images from cameras to find the location of cubes around the objects in 3D space. This task requires synchronized data streams from a LiDAR and one or more cameras. There are several approaches to fuse the sparse data from the LiDAR with colour images like these ones [12] [13].

A number of successful multi-class object detection techniques have appeared in the recent years, thanks to the re-birth of neural networks, in the form of deep learning models, such as Faster R-CNN [14] or MobileNet [15], which are able to provide extraordinary detection accuracy and robustness and operate real-time in GPU-enabled platforms. For instance, experiments carried out in Nvidia DrivePX2 having a dGPU with 1152 cuda cores, we obtained inference times from 20-35ms per 3Mpixels image using different SSD and Faster-RCNN models. Other Deep Learning models for 2D object detection in the literature [11] have reported times between

ISBN 978-90-827970-1-5 © EURASIP 2018

471 to 1476ms in Nvidia K40 (2880 cuda cores), between 80 to 300ms in Nvidia Titan X (3072 cores) and up to 4200ms in Nvidia GTX 1050 (256 cores). These values depend on the complexity of the neural network architecture and the size of their input layers.

Scene semantic segmentation

This is a fundamental topic in computer vision in which each pixel in the image is assigned a label from a predefined set of categories [16]. They typically consist of road, sidewalk, sky, vegetation, traffic signs, people, vehicles, etc. A sample image in Figure 1 illustrates the case. Given the natural complexity of urban scenes and depending on the number of selected classes, this task requires high computation resources not only during training to learn the patterns but also during inference due to its pixelwise design. In the state of the art, DNN approaches reported per frame delays between 33ms to 35seconds for Nvidia Titan X, depending on the proposed model [11].

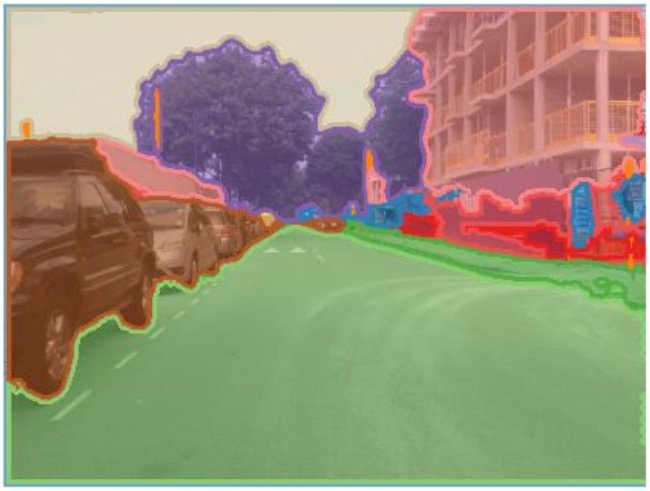

Figure 1: Example of semnatic segmentation

Visual odometry

In visual odometry the goal is to recover the full trajectory of a camera system from images. A relative transformation is estimated to represent the vehicle motion from one-time step to the next one. This is achieved by registering two consecutive frames based on matched salient features. The current state of the art in this task can obtain good results in standard CPUs without requiring further processing capabilities [17].

Stereo reconstruction

Using a pair of cameras in a stereo setup, depth information can be recovered from the 2D images, thus achieving a 3D reconstruction of the scene. This process requires a good camera synchronization and a previous calibration of the stereo rig. Then, an expensive pixel-based or region-based algorithm obtains the disparity between the images. The algorithm finds correspondences between the two views at the same point in time which are the projections of the same physical surface in the 3D world. The challenge is to obtain a disparity map as dense as possible to increase 3D accuracy and this poses important computing requirements [18]. For instance, different algorithms take from 410 to 1300ms in Nvidia Titan X and 470ms in Nvidia GTX 1080 (2560 cores).

Driver monitoring

Fully autonomous cars will potentially require few participation of humans in the driving tasks. However, as the technology transitions to that final goal of autonomous vehicles, there is still the need to monitor the attention state of the driver [19]. Nevertheless, specific scenarios will require driver engagement. Image-based systems are one of the best options for monitoring the distraction and fatigue level of the driver[20]. Usually these systems capture images from the face (see Figure 2) and body and process them in two steps. First, several relevant features are computed such as eyelid activity, eye gaze direction, head pose or upper body pose. Then, the driver attention state is evaluated using these features. Modern driver monitoring systems are based on different machine learning techniques, including deep learning architectures such as MobileNet [14], SSD [21] and ResNet [22] to extract the physiological features of the driver and extract data patterns that indicate the attention level.

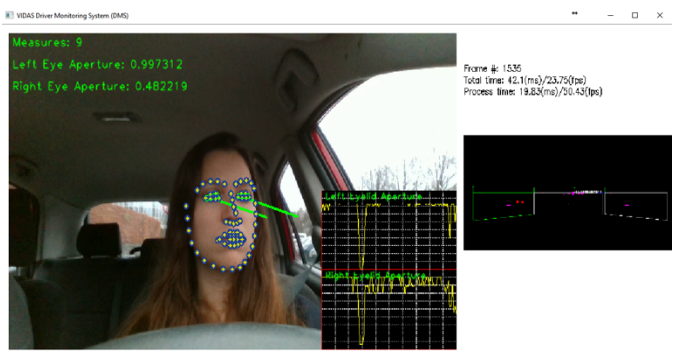

Figure 2: Example of DMS measurements

Numerical examples of GPU requirements per ADAS application

Table 1 gives some numerical examples from our experience on running ADAS applications on GPU hardware. The table also highlights the network requirements needed in order to support a specific application. The compressed size of the network stream was calculated using the height x weight x frame_rate x motion_factor x 0.07, as it's the case for the H264 format. The motion factor value was 4, which is the suitable for video input with high motion elements, such as sports or, in our case, vehicles moving at speed.

ISBN 978-90-827970-1-5 © EURASIP 2018

Table 1: GPU requirements versus execution time and network utilization

Application	Image size	Time per frame	Processing	Stream size uncompressed (Mbps)	Stream size compressed (Mbps)
2D Object detection	0.5 Mpix colour	30 – 4200 ms	GPU cores 3072 to 640	300	3.5
Multi-Lane detection	0.5 Mpix colour	6 – 1200 ms	GPU cores 3072 to 640	300	3.5
Scene semantic segmentation	2Mpix colour	60 – 35000 ms	GPU cores 3072 to 640	1,200	14
3D object detection	0.5 Mpix image + 1M 3D points	300 – 1000 ms	GPU cores 3072 to 640	396	99.5
Stereo reconstruction	2x0.5 Mpix	470 – 1300 ms	GPU cores 3072 to 2560	600	7
Driver monitoring	0.3 Mpix	13 – 70 ms	GPU cores 3584 to 256	180	2.1

IV. Numerical analysis

Table 1 describes the relationship between runtimes and processing hardware requirements as reported in the literature and some of them also measured in our experiments. Typically, the higher number of GPGPU cores, the lower processing times. However, literature in Deep Learning applied to ADAS is broad and many other aspects can influence the delays: complexity and combinations of neural network architectures, size of input layer, software platform (Caffe, Tensorflow, Theano), network and/or hardware optimizations, coding language (python vs C++), etc.

A. Parallel performance

As observed in Table 1, the execution time for certain computer vision automotive applications can be small, which would qualify as real-time image processing. A typical threshold in the literature [2] is 30 to 40 milliseconds processing time per frame, at a 25 frames per second video rate. However, the execution time is mainly influenced by the number of GPUs available for processing. Figure 3 depicts the applications highlighted in Table 1 that rely on GPUs for image analysis. Let us consider the slowest processing time given the best hardware available is used, which in our case is stereo reconstruction. The minimum time to process a frame for this application is 470 milliseconds and an algorithm using fused information from all video cameras and application can only take decisions as fast as the slowest feed, which is 470 milliseconds in our case. Given that the execution time depends on the number of GPU cores that we offer to a specific application, we took an approach to determine the number of cores that can be released and reused in other applications, so as to delay the execution of other applications up to the slowest application that runs on the best hardware.

We can thus observe from Figure 3 that more than 1000 cores can be freed up and offered to other applications as the best parallel performance is limited by the slowest performing application. The calculations have been done with 3072 cores available to each application and that would yield the application's fastest execution time, while the slowest performance is computed at 640 cores per application. The number of cores needed by a delayed application is calculated through linear interpolation, and depicted in Figure 3.

B. Network requirements for cloud offload

We move now to discussing the implications and feasibility of transferring some of the video feeds to external systems where

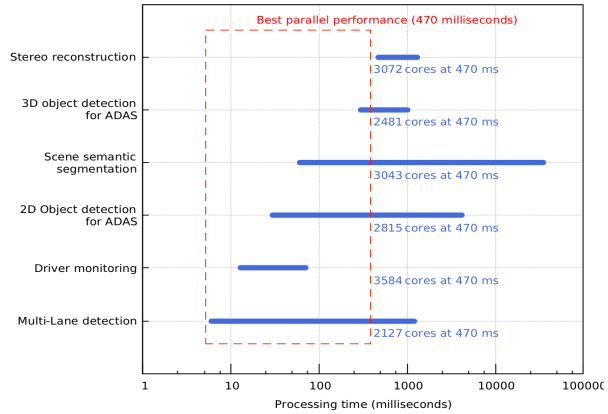

Figure 3: Processing time versus ADAS application. (Note: x-axis is logscale)

processing is scaled up more easily and running costs are smaller. One such external system is the cloud, where current research trends have triggered cloud providers to offer GPUs not only for video rendering but also for running neural networks for machine learning algorithms. In this work we investigate the network demands of the applications in Table 1. We considered the applications that need GPU for video processing and considered the number of cores needed when all these applications run on 3072 cores GPUs in the vehicle. The question we want to answer is how many of these cores are actually needed, given that the video content can be uploaded to the cloud where processing is cheaper and faster, however the size of the video feeds poses a real problem. In Table 1 we observe the transfer rates required for both uncompressed and compressed video feeds. The reason we considered the compressed solution is that the combined size of the video feeds is considerably larger than the capabilities of current wireless and cellular technology. We show in Figure 4 that the total number of GPUs required to run all the applications is over 18944, and the figure also depicts the number of cores that can be offloaded to the cloud, for uncompressed video feeds. We have considered going up to 1Gbps upload speed for a connected vehicle, which can be achieved in theory by 5G technology. The figure shows that even at a high upload speed, only 2 applications can be offloaded to the cloud for processing, and that without considering the possible delays that can affect the local decision making.

Figure 5 shows that the cloud can be really considered as an offload solution as the required upload speeds are an order of magnitude smaller than in the case of uncompressed video.

ISBN 978-90-827970-1-5 © EURASIP 2018

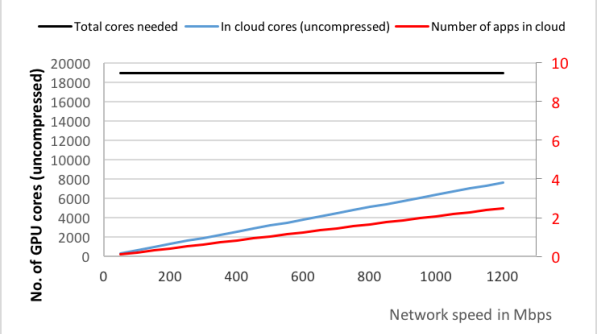

Figure 4: Number of GPU cores that the cloud can support versus the speed of the vehicle's network connection transferring uncompressed video content. Secondary y axis shows number of ADAS applications that the vehicle can offload to the cloud (i.e. red line).

With speeds of 130 Mbps all 5 ADAS applications can have their associated video feeds transferred to the cloud and analysed there by GPUs. There are some drawbacks when using compressed video, however where sufficient network resources are guaranteed to vehicles, Figure 5 shows that it is feasible to consider video processing offloading to the cloud.

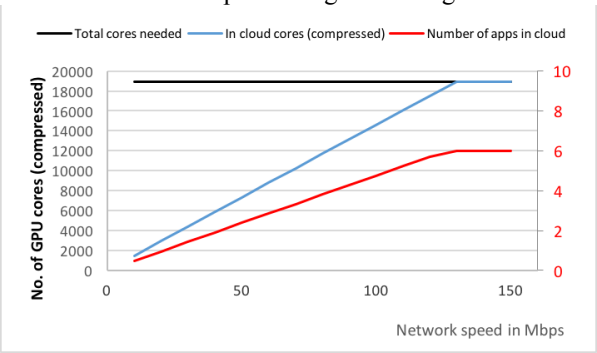

Figure 5: Number of GPU cores that the cloud can support versus the speed of the vehicle's network connection transferring compressed video content. Secondary y axis shows number of ADAS applications that the vehicle can offload to the cloud (i.e. red line)

Network latency can become another factor in the decision, and current research trends showed that could-like compute power can be brought closer to the end users by employing the Edge Computing paradigm[3], [4], [6]. In case of utilizing a cellular network infrastructure for connectivity, then the mobile network architecture of the future will incorporate Edge Computing functions [6]. This will have a great impact on reducing the latency of the connection and some research [6] even considered offloading decision-making to servers running on the edge of the cellular network, at one-hop network distance from the actual vehicle.

V. Conclusion

This paper focused on ADAS applications which involve the aid of computer vision breakthroughs for processing a large volume of video input captured from cameras capturing both the interior and the surrounding environment of a vehicle. These applications need GPUs for near real-time processing, so as to allow the output of the processing to be used by self-driving algorithms. The paper gives examples of ADAS applications and performance characteristics and we showed that with the help of 5G speeds some of the ADAS applications can be offloaded to the cloud. This is an ongoing investigation, and our future work will focus on incorporating more factors such as network latency and more varied benchmarks for computer vision performance.

Acknowledgment

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 690772, project VI-DAS).

References

[1] J. Fung, "Computer Vision on the GPU," *GPU Gems*, vol. 2, pp. 649–666, 2005.
[2] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art," 2017.
[3] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE 2016*, 2016.
[4] D. J. Deng, S. Y. Lien, C. C. Lin, S. C. Hung, and W. B. Chen, "Latency Control in Software-Defined Mobile-Edge Vehicular Networking," *IEEE Commun. Mag.*, vol. 55, no. 8, 2017.
[5] S. Nunna *et al.*, "Enabling Real-Time Context-Aware Collaboration through 5G and Mobile Edge Computing," *Proc. - 12th Int. Conf. Inf. Technol. New Gener. ITNG 2015*, pp. 601–605, 2015.
[6] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, 2017.
[7] J. Javier Yebes, L. M. Bergasa, and M. Á. García-Garrido, "Visual object recognition with 3D-aware features in KITTI urban scenes," *Sensors (Switzerland)*, vol. 15, no. 4, pp. 9228–9250, 2015.
[8] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[9] I. T. U. Journal, I. C. T. Discoveries, and S. I. No, "RECONFIGURABLE PROCESSOR FOR DEEP LEARNING IN AUTONOMOUS VEHICLES Deephi Tech , Beijing , China Institute of Microelectronics , Tsinghua University , Beijing , China Department of Electrical Engineering , Stanford University , Stanford CA , USA," no. 1, pp. 1–13, 2017.
[10] M. Nieto, A. Cortés, O. Otaegui, J. Arróspide, and L. Salgado, "Real-time lane tracking using Rao-Blackwellized particle filter," *J. Real-Time Image Process.*, vol. 11, no. 1, pp. 179–191, 2016.
[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
[12] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," 2016.
[13] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks," 2016.
[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
[15] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.
[16] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," 2016.
[17] J. Zhang and S. Singh, "Visual-lidar Odometry and Mapping: Low-rift, Robust, and Fast Localization," *Icra*, pp. 393–398, 2015.
[18] Z. Liang *et al.*, "Learning Deep Correspondence through Prior and Posterior Feature Constancy," 2017.
[19] J. Gonçalves and K. Bengler, "Driver State Monitoring Systems– Transferable knowledge Manual driving to HAD," in *Proceedings of the 6th International Conference on Applied Human Factors and Ergonomics*, 2015, vol. 3, no. Ahfe, pp. 3011–3016.
[20] Y. Dong, Z. Hu, K. Uchimura, and N. Murayama, "Driver inattention monitoring system for intelligent vehicles: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 596–614, 2011.
[21] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.
[22] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimed. Tools Appl.*, pp. 1–17, 2017.