INVITED PAPER

Edge Computing for Autonomous Driving: Opportunities and Challenges

This article surveys the designs of autonomous driving edge computing systems. In addition, it presents the security issues in autonomous driving as well as how edge computing designs can address these issues.

By Shaoshan Liu, *Senior Member IEEE*, Liangkai Liu, Jie Tang, *Member IEEE*, Bo Yu, *Member IEEE*, Yifan Wang, and Weisong Shi, *Fellow IEEE*

ABSTRACT | Safety is the most important requirement for autonomous vehicles; hence, the ultimate challenge of designing an edge computing ecosystem for autonomous vehicles is to deliver enough computing power, redundancy, and security so as to guarantee the safety of autonomous vehicles. Specifically, autonomous driving systems are extremely complex; they tightly integrate many technologies, including sensing, localization, perception, decision making, as well as the smooth interactions with cloud platforms for high-definition (HD) map generation and data storage. These complexities impose numerous challenges for the design of autonomous driving edge computing systems. First, edge computing systems for autonomous driving need to process an enormous amount of data in real time, and often the incoming data from different sensors are highly heterogeneous. Since autonomous driving edge computing systems are mobile, they often have very strict energy consumption restrictions. Thus, it is imperative to deliver sufficient computing power with reasonable energy consumption, to guarantee the safety of autonomous vehicles, even at high speed. Second, in addition to the edge system design, vehicle-to-everything (V2X) provides redundancy for autonomous driving workloads and alleviates stringent performance and energy constraints on the edge side. With V2X, more research is required to define how vehicles cooperate with each other and the infrastructure. Last, safety cannot be guaranteed when security is compromised. Thus, protecting autonomous driving edge computing systems against attacks at different layers of the sensing and computing stack is of paramount concern. In this paper, we review state-of-the-art approaches in these areas as well as explore potential solutions to address these challenges.

KEYWORDS | Connected and autonomous vehicles (CAVs); edge computing; heterogeneous computing; security; vehicle-to-everything (V2X); vehicular operating system.

I. INTRODUCTION

The design goal of autonomous driving edge computing systems is to guarantee the safety of autonomous vehicles. This is extremely challenging, as autonomous vehicles need to process an enormous amount of data in real time (as high as 2 GB/s) with extremely tight latency constraints [1]. For instance, if an autonomous vehicle travels at 60 miles per hour (mph), and thus about 30 m of braking distance, this requires the autonomous driving system to predict potential dangers up to a few seconds before they occur. Therefore, the faster the autonomous driving edge computing system performs these complex computations, the safer the autonomous vehicle is.

Specifically, autonomous driving systems are extremely complex; they tightly integrate many technologies, including sensing, localization, perception, decision making, as well as the smooth interaction with cloud platforms for high-definition (HD) map generation and data storage [2].

These complexities pose many challenges for the design of autonomous driving edge computing systems, just to

Manuscript received February 10, 2019; revised April 6, 2019; accepted May 5, 2019. Date of publication June 24, 2019; date of current version August 5, 2019. *(Corresponding author: Jie Tang.)*

S. Liu and **B. Yu** are with PerceptIn, Fremont, CA 94539 USA.

L. Liu and **W. Shi** are with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA.

J. Tang is with the School of Computer Science & Engineering, South China University of Technology, Guangzhou 510330, China (e-mail: cstangjie@scut.edu.cn).

Y. Wang is with Wayne State University, Detroit, MI 48202 USA, and also with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China.

Digital Object Identifier 10.1109/JPROC.2019.2915983

name a few. First, they need to enable the interactions between the functional modules with low overheads, and this requires a lightweight operating system. Second, they need to process an enormous amount of data in real time, and often the incoming data from different sensors are highly heterogeneous. In addition, since they are mobile systems, they often have very strict energy consumption restrictions. This requires a high-performance and energy-efficient hardware system.

In addition to the edge system design, vehicle-to-everything (V2X) technologies should also get involved in the communication of the edge system. V2X provides redundancy for autonomous driving workloads; it also alleviates stress on the edge computing system. We believe this is a promising approach, but the key is to identify a sweet spot between the tradeoffs of fully relying on the edge computing system versus fully relying on the V2X infrastructure. Hence, exploring how V2X enables autonomous vehicles to cooperate with each other and with the infrastructure remains an open research topic.

Having a high-performance and energy-efficient edge computing system is not enough, as attackers may turn an autonomous vehicle into a dangerous weapon by hacking into any layer of the autonomous driving sensing and computing stack. More research is required to identify all possible attack surfaces as well as protection mechanisms before autonomous vehicles are released on public roads.

To summarize, the overarching challenge of autonomous driving edge computing systems design is to efficiently integrate the functional modules, including interactions with edge servers and V2X infrastructure, to process a massive amount of heterogeneous data in real time, within a limited energy budget, and without sacrificing the security of the users. In this paper, we review state-of-the-art approaches in these areas as well as explore potential solutions to address these challenges.

Before delving into the details of the autonomous driving edge computing system designs, let us first define the autonomous driving computing ecosystem. As illustrated in Fig. 1, each autonomous vehicle is equipped with an edge computing system, which integrates all the real-time functional modules, such as localization, perception, planning and control, and so on. Then, each vehicle communicates with edge servers, and eventually with the central cloud, through existing 3G/4G/5G communication networks. In addition, the vehicles can communicate with the road side units (RSUs) through either the 5G networks or the dedicated short-range communications (DSRC) networks; this is a typical vehicle-to-infrastructure (V2I) scenario. Moreover, the vehicles can communicate with each other through the DSRC networks, and this is a typical vehicle-to-vehicle (V2V) scenario.

The remainder of this paper is organized as follows. In Section II, we review the background of autonomous driving technologies and help readers understand the complexities of autonomous driving systems. In Section III, the latest progress on the designs of autonomous driving edge computing systems is presented and discussed. In Section IV, we focus on how V2X alleviates the stress on edge computing systems. In Section V, we review security issues in autonomous driving and how edge computing designs address these issues. We conclude in Section VI.

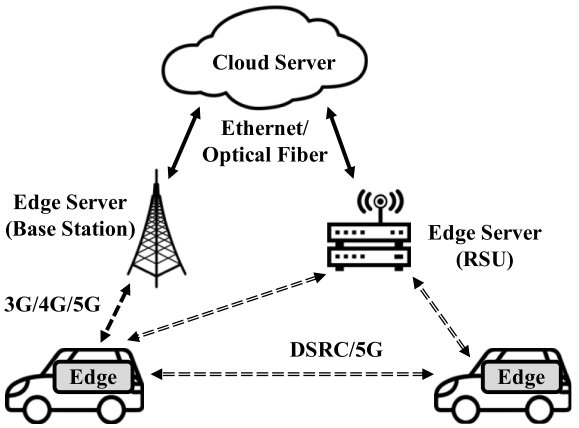

Fig. 1. *Autonomous driving edge computing architecture.*

II. AUTONOMOUS DRIVING TECHNOLOGIES

As detailed in [3], autonomous driving is not one technology but rather an integration of many technologies. As is shown in Fig. 2, the autonomous driving technology stack consists of three major subsystems: algorithms, including sensing, perception, and decision; a vehicular edge subsystem, including the operating system and hardware platform; and a cloud platform, including data storage, simulation, HD mapping, and deep learning model training.

A. Algorithm Subsystem

The algorithm subsystem extracts meaningful information from sensor raw data to understand its environment and dynamically make decisions about its actions.

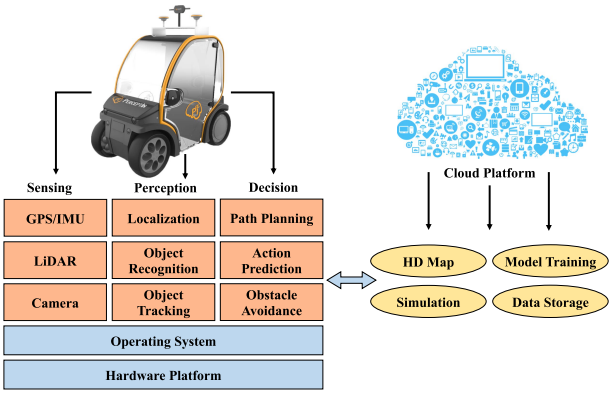

Fig. 2. *Autonomous driving technology stack.*

1) Sensing: Normally, an autonomous vehicle consists of several major sensors. Since each type of sensor presents advantages and drawbacks, the data from multiple sensors must be combined. The sensor types include the following.

a) GNSS/IMU: The global navigation satellite system and inertial measurement unit (GNSS/IMU) system helps autonomous vehicles localize themselves by reporting both inertial updates and a global position estimate at a high rate, e.g., 200 Hz. While GNSS is a fairly accurate localization sensor at only 10 Hz, its update rate is too slow to provide real-time updates. Now, although an IMU's accuracy degrades with time and thus cannot be relied upon to provide accurate position updates over long periods, it provides updates more frequently—at, or higher than, 200 Hz. This should satisfy the real-time requirement. By combining GNSS and IMU, we provide accurate and real-time updates for vehicle localization.

b) LiDAR: Light detection and ranging (LiDAR) is used for mapping, localization, and obstacle avoidance. It works by bouncing a beam off surfaces and measuring the reflection time to determine distance. LiDAR is used to generate HD maps, localize a moving vehicle against HD maps, detect obstacles ahead, and so on.

c) Cameras: Cameras are mostly used for object recognition and object tracking tasks, such as lane detection, traffic light detection, pedestrian detection, and more. To enhance autonomous vehicle safety, existing implementations usually mount eight or more cameras around the car, and we use cameras to detect, recognize, and track objects in front, behind, and on both sides of the vehicle. These cameras usually run at 60 Hz and, when combined, generate around multiple gigabytes of raw data per second.

d) Radar and Sonar: The radar and sonar system is used for the last line of defense in obstacle avoidance. The data generated by radar and sonar show the distance from the nearest object in front of the vehicle's path. When we detect that an object is not far ahead and that there may be a danger of a collision, the autonomous vehicle should apply the brakes or turn to avoid the obstacle. Therefore, the data generated by radar and sonar do not require much processing and are usually fed directly to the control processor—not through the main computation pipeline—to implement such urgent functions as swerving and applying the brakes.

2) Perception—Localization: The perception subsystem consumes incoming sensor data to understand the vehicle's environment. The main tasks in autonomous driving perception include localization, object detection, object tracking, and so on.

While GNSS/IMU is used for localization, GNSS provides fairly accurate localization results but with a slow update rate; IMU provides a fast update with less accurate results. We use Kalman filtering to combine the advantages of the two and provide accurate and real-time position updates [4]. Nonetheless, we cannot solely rely on this combination for localization for three reasons: 1) the accuracy is only about one meter; 2) the GNSS signal has multipath problems, which means that the signal may bounce off buildings and introduce more noise; and 3) GNSS requires an unobstructed view of the sky so it fails to work in closed environments such as tunnels.

Cameras are used for localization. The pipeline line of Vision-based localization is simplified as follows: 1) by triangulating stereo image pairs, a disparity map is obtained and used to derive depth information for each point; 2) by matching salient features between successive stereo image frames in order to establish correlations between feature points in different frames, the motion between the past two frames is estimated; and 3) by comparing the salient features against those in the known map, we derive the current position of the vehicle [5].

LiDAR is used for localization, relying heavily on a particle filter [6]. The point clouds generated by LiDAR provide a "shape description" of the environment, but it is hard to differentiate individual points. By using a particle filter, the system compares a specific observed shape against the known map to reduce uncertainty. To localize a moving vehicle relative to these maps, we apply a particle filter method to correlate the LiDAR measurements with the map. The particle filter method has been demonstrated to achieve real-time localization with 10-cm accuracy and is effective in urban environments.

3) Perception—Object Recognition and Tracking: In recent years, we have seen the rapid development of deep learning technology, which achieves significant object detection and tracking accuracy. A convolution neural network (CNN) is a type of deep neural network (DNN) that is widely used in object recognition tasks. A general CNN evaluation pipeline usually consists of the following layers: 1) the convolution layer uses different filters to extract different features from the input image. Each filter contains a set of "learnable" parameters that will be derived after the training stage; 2) the activation layer decides whether to activate the target neuron; 3) the pooling layer reduces the spatial size of the representation to reduce the number of parameters and consequently the computation in the network; and last, 4) the fully connected layer connects all neurons to all activations in the previous layer [7].

Once an object is identified using a CNN, next comes the automatic estimation of the trajectory of that object as it moves—or, object tracking. Object tracking technology is used to track nearby moving vehicles, as well as people crossing the road, to ensure the current vehicle does not collide with moving objects. In recent years, deep learning techniques have demonstrated advantages in object tracking compared to conventional computer vision techniques. By using auxiliary natural images, a stacked autoencoder is trained offline to learn generic image features that are more robust against variations in viewpoints and vehicle

positions. Then, the offline-trained model is applied for online tracking [8].

4) Decision: In the decision stage, action prediction, path planning, and obstacle avoidance mechanisms are combined to generate an effective action plan in real time. One of the main challenges for human drivers when navigating through traffic is to cope with the possible actions of other drivers, which directly influences their own driving strategy. This is especially true when there are multiple lanes on the road or at a traffic change point.

To make sure that autonomous vehicles travel safely in these environments, the decision unit generates predictions of nearby vehicles and then decides on an action plan based on these predictions. To predict the actions of other vehicles, we generate a stochastic model of the reachable position sets of the other traffic participants and associate these reachable sets with probability distributions [9].

Planning the path of an autonomous, responsive vehicle in a dynamic environment is a complex problem, especially when the vehicle is required to use its full maneuvering capabilities. One approach would be to use deterministic, complete algorithms to search all possible paths and utilize a cost function to identify the best path. However, this requires enormous computational resources and may be unable to deliver real-time navigation plans. To circumvent this computational complexity and provide effective real-time path planning, probabilistic planners have been utilized [10].

Since safety is of paramount concern in autonomous driving, we should employ at least two levels of obstacle avoidance mechanisms to ensure that the vehicle will not collide with obstacles. The first level is proactive and based on traffic predictions. The traffic prediction mechanism generates measures like time-to-collision or predicted-minimum-distance. Based on these measures, the obstacle avoidance mechanism is triggered to perform local path replanning. If the proactive mechanism fails, the second-level reactive mechanism, using radar data, takes over. Once the radar detects an obstacle ahead of the path, it overrides the current controls to avoid the obstacle [11].

B. Vehicular Edge Subsystem

The vehicular edge subsystem integrates the above-mentioned algorithms together to meet real-time and reliability requirements. There are three challenges to overcome: 1) the system needs to make sure that the processing pipeline is fast enough to consume the enormous amount of sensor data generated; 2) if a part of the system fails, it needs to be robust enough to recover from the failure; and 3) the system needs to perform all the computations under energy and resource constraints.

1) Hardware Architecture: First, we review the existing computer architecture for autonomous driving tasks. A more detailed review is found in [1]. The Nvidia PX platform is the current leading GPU-based solution for autonomous driving [12]. Each PX 2 consists of two Tegra systems-on-chip (SoCs) and two Pascal graphics processors. Each GPU has its own dedicated memory, as well as specialized instructions for DNN acceleration. To deliver high throughput, each Tegra connects directly to the Pascal GPU using a PCI-E Gen 2 $\times$ 4 bus (total bandwidth: 4.0 GB/s). In addition, the dual CPU-GPU cluster is connected over the gigabit Ethernet, delivering 70 Gb/s. With optimized I/O architecture and DNN acceleration, each PX2 is able to perform 24 trillion deep-learning calculations every second.

Texas Instruments' TDA provides a DSP-based solution for autonomous driving. A TDA2x SoC consists of two floating-point C66x DSP cores and four fully programmable vision accelerators, which are designed for vision processing functions. The vision accelerators provide eight-fold acceleration on vision tasks compared to an ARM Cortex-15 CPU while consuming less power [13]. Similarly, CEVA XM4 is another DSP-based autonomous driving computing solution. It is designed for computer vision tasks on video streams. The main benefit for using CEVA-XM4 is energy efficiency, which requires less than 30 mW for a 1080p video at 30 frames/s [14].

Altera's Cyclone V SoC is one field-programmable gate array (FPGA)-based autonomous driving solution, which has been used in Audi products. Altera's FPGAs are optimized for sensor fusion, combining data from multiple sensors in the vehicle for highly reliable object detection [15]. Similarly, Zynq UltraScale MPSoC is also designed for autonomous driving tasks [16]. When running CNN tasks, it achieves 14 images/s/W, which outperforms the Tesla K40 GPU (4 images/s/W). Also, for object tracking tasks, it reaches 60 frames/s in a live 1080p video stream.

MobilEye EyeQ5 is a leading application-specified integrated circuit (ASIC)-based solution for autonomous driving [17]. EyeQ5 features heterogeneous, fully programmable accelerators, where each of the four accelerator types in the chip is optimized for their own family of algorithms, including computer-vision, signal-processing, and machine-learning tasks. This diversity of accelerator architectures enables applications to save both computational time and energy by using the most suitable core for every task. To enable system expansion with multiple EyeQ5 devices, EyeQ5 implements two PCI-E ports for interprocessor communication.

2) Real-Time Operating Systems: A real-time operating system (RTOS) is a special operating system intended to serve real-time applications, such as industrial, automotive, aviation, military, and so on [18]. QNX is a popular commercial RTOS widely used in the automotive industry. The QNX kernel contains only CPU scheduling, interprocess communication, interrupt redirection, and timers. Everything else runs as a user process, including a special process known as proc that performs process creation

and memory management by operating in conjunction with the microkernel [19]. This is achieved by two key mechanisms—subroutine-call-type interprocess communication and a boot loader that loads an image which contains not only the kernel but also any desired collection of user programs and shared libraries. There are no device drivers in the kernel.

Another popular commercial RTOS is VxWorks, which is designed for use in embedded systems requiring real-time, deterministic performance and, in many cases, safety and security certification [20]. VxWorks supports multiple architectures including Intel architecture, POWER architecture, and ARM architectures. VxWorks have been used in multicore asymmetric multiprocessing, symmetric multiprocessing, and mixed modes.

Both QNX and VxWorks use real-time kernels for mission-critical applications subject to real-time constraints, which guaranty a response within predefined time constraints. While QNX is based on a message passing architecture, VxWorks utilizes a shared memory architecture. Message passing is fundamental to the kernel design, which allows the system to pass information from one task to another or to several others in the system. Shared memory architecture refers to a system that has its own private address space for physically distributed memories.

On kernel design, QNX uses a microkernel, and VxWorks uses a monolithic kernel. A microkernel leverages system calls to manage basic services such as address space management, thread management, and interprocess communications. A monolithic kernel manages all the basic services and user-defined services including interprocess communications in protected kernel space. Therefore, using a monolithic kernel design, VxWorks is self-contained.

On priority scheduling under QNX, all the processes run on a priority-driven preemptive basis, meaning the process with the highest priority gets to access the CPU first and the priorities range from 0 to 31. The scheduling occurs in real time, and every thread inherits its parent's priority by default. VxWorks, on the other hand, uses only two types of scheduling algorithms, preemptive priority-based and round-robin scheduling.

3) Middleware: On top of the RTOS, we need a middleware layer to bind different autonomous driving services together. To achieve this, most existing autonomous driving solutions utilize the robot operating system (ROS) [21] or modified versions of ROS. Specifically, ROS is a communication middleware that facilitates communications between different parts of an autonomous vehicle system. For instance, the image capture service publishes messages through ROS, and both the localization service and the obstacle detection service subscribe to the published images to generate position and obstacle updates.

Although ROS is a popular choice for the operating system in autonomous vehicle systems, in practice, it suffers from a few problems. The first one is reliability; ROS has a single master and no monitor to recover failed nodes.

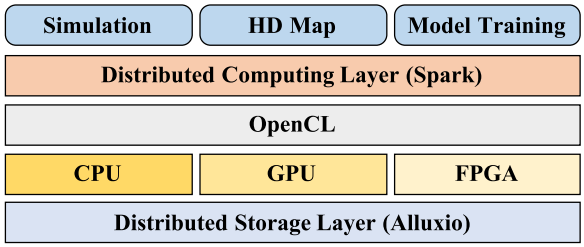

Fig. 3. *Autonomous driving cloud architecture.*

Performance is second; when sending out broadcast messages, it duplicates the message multiple times, leading to performance degradation. Security is also a concern; it has no authentication and encryption mechanisms. Although ROS 2.0 promised to fix these problems, it has not yet been extensively tested, and many features are not yet available. Most importantly, for a piece of software to run securely and reliably in vehicles, it needs to meet certain automotive-grade standards, such as ISO26262 [22]. Unfortunately, an automotive-grade middleware is still missing today.

C. Cloud Subsystem

Autonomous vehicles are mobile systems that, therefore, need a cloud subsystem to benefit from the extra computing power of a cloud subsystem. The two main functions provided by the cloud include distributed computing and distributed storage. Such a cloud subsystem has several applications, including simulation, which is used to verify new algorithms, HD map production, and deep learning model training [23]. Note that due to the limitations on communication bandwidth, latency, and reliability, currently, we only perform offline computing tasks on the cloud. With more advancements in communication infrastructure, we enable more edge-cloud cooperation in real-time computing tasks.

1) Simulation: The first application of a cloud platform system is a simulation. There are two main kinds of simulation technologies. The first type is to simulate the environment based on synthetic data; these simulators are mainly used for the control and planning, especially at the initial development stage of the algorithm. The second type of simulator is based on real data playback to verify the function and performance of different components, which is mainly used in the iterative process of algorithm development. Take planning and control algorithm development for example, with offline simulations, we verify the effectiveness of planning and control algorithms, especially with different corner cases. Then, once the algorithms pass all the simulation scenarios, we deploy the algorithms in a vehicle for real-time execution.

As described in [24], the core problem of the simulator lies in how realistically we simulate the actual driving environment. No matter how good the simulator

is, the artificial simulation of the scene and the real scene still has some differences. There are still many unexpected events in the real scene that cannot be simulated in a simulator. Therefore, if real traffic data are used to reproduce the real scene, better test results will be achieved compared to the artificial simulation of the scene.

However, the major problem of replaying real-world data is the computing power required to process it. In order to reproduce the scene of every section of the real world on the simulator, the autonomous vehicles need to collect the information of each section of the road. This amount of data cannot be processed on single machines. In addition, in each scene, it is further broken down into basic fragments and rearrange the combinations of these fragments to generate more test cases. However, this would generate even more data and add more burden to the simulation platform which is already stressed.

2) HD Map Generation: As detailed in [25], HD maps are generated in the following steps.

Pose Estimation: For map data collection vehicles, getting accurate poses (location and orientation) of the vehicles is key to generating HD maps. If the poses of the vehicles collecting map data are inaccurate, it is impossible to produce precise maps. Once we have the accurate poses of the data-collecting vehicles, and with accurate sensor installation information such as how the sensors are mounted and their relative angles to the vehicle frame, we could calculate the accurate poses of the generated point clouds.

Map Data Fusion: Once we have accurate poses, the next step is map data fusion. Map data here includes LiDAR 3-D point clouds and camera images. In this step, multiple scans of point clouds are calibrated and then aligned to generate denser point clouds, and the generated point clouds and captured images are then registered to each other. This way, we could use the point clouds to get the 3-D location of objects and use the registered images to extract semantic information. Note that, point clouds accurately provide 3-D positions but usually do not allow semantic information extraction, while on the other hand, images are great for semantic information extraction. By combining point clouds and captured images, we allow both accurate poses extraction and semantic information extraction.

Three-Dimensional Object Location Detection: For basic road elements whose geometry and precise locations are important (e.g., lane boundaries, curbs, traffic lights, overpasses, railway tracks, guardrails, light poles, speed bumps, even potholes, etc.), we need to register their precise 3-D locations and geometries in the HD maps.

Semantics/Attribute Extraction: At last, we extract semantics and attributes from captured data and integrate these into the HD Maps. The semantic information includes lane/road model construction, traffic signs recognition and their association with lanes, the association of traffic lights with lanes, road marking semantics extraction, road elements, and so on.

3) Model Training: The second application this infrastructure needs to support is offline model training. As we use different deep learning models in autonomous driving, it is imperative to provide updates that will continuously improve the effectiveness and efficiency of these models. However, since the amount of raw data generated is enormous, we would not be able to achieve fast model training using single servers.

One effective way to achieve high performance is through the utilization of heterogeneous computing and high-throughput storage systems. As described in [23], Liu *et al.* integrated a distributed computing platform, a heterogeneous computing acceleration layer, and a distributed storage layer together to deliver an extremely high-performance and high-throughput distributed training engine. Specifically, to synchronize the nodes, at the end of each training iteration, this system summarizes all the parameter updates from each node, performs calculations to derive a new set of parameters, and then broadcasts the new set of parameters to each node so they start the next iteration of training. It is the role of the parameter server to efficiently store and update the parameters.

III. INNOVATIONS ON THE VEHICULAR COMPUTING EDGE

In this Section, we review the latest progress in the design of edge computing systems for autonomous driving applications. First, we start with benchmark suites available for evaluating edge computing system designs. Second, we review different approaches in designing computer architectures for autonomous driving workloads. Third, we describe the designs of runtime layers for efficient mapping of incoming workloads onto heterogeneous computing units. Fourth, we discuss the designs of middleware for binding different autonomous driving functional modules. Last, we present three examples of autonomous driving edge computing systems.

A. Benchmark Tools

To improve a computing system, the most effective tool is a standard benchmark suite to represent the workloads widely used in the target applications. The same principle applies when it comes to designing and improving edge computing systems for autonomous vehicles. Current research in this area is divided into two categories: data sets and workloads. KITTI [26], [27] was the first benchmark data set related to autonomous driving. It is composed of rich vision sensor data with labels, such as monocular/stereo image data and 3-D LiDAR data. According to different data types, it also provides a dedicated method to generate the ground truth and to calculate the evaluation metrics. KITTI was built for evaluating the performance of algorithms in the autonomous driving scenario, including but not limited to visual odometry, lane detection, object detection, and object tracking.

In addition to KITTI, there are some customized benchmark data sets for each algorithm, such as the Technical University of Munich Red Green Blue-Depth (TUM RGB-D) [28] for RGB-D simultaneous localization and mapping (SLAM), PASCAL3D [29] for 3-D object detection, and the MOTChallenge benchmark [30], [31] for multitarget tracking. These kinds of data sets serve as very good data sources for stressing edge computing systems.

Another class of related benchmark suites is designed to benchmark the performance of novel hardware architectures and software framework, which usually consists of a set of computer vision kernels and applications. The San Diego Vision Benchmark Suite (SD-VBS) [32] and MEVBench [33] both are performance benchmark suites for mobile computer vision system. SD-VBS provides single-threaded C and MATLAB implementations of nine high-level vision applications. MEVBench is an extended benchmark based on SD-VBS. It provides single-threaded and multithreaded C++ implementations of 15 vision applications. However, these two benchmarks are prior works in the field, so they are not targeted toward heterogeneous platforms such as GPUs and did not contain novel workloads, such as the deep learning algorithms. SLAMBench [34] concentrates on using a complete RGB-D SLAM application to evaluate novel heterogeneous hardware. It takes KinectFusion [35] as the implementation and provided C++, OpenMP, OpenCL, and CUDA versions of key function kernels for heterogeneous hardware. These efforts are a step in the right direction, but we still need a comprehensive benchmark that contains diverse workloads that cover varied application scenarios of autonomous vehicles (like the MAVBench [126] for micro aerial vehicle system benchmarking) to evaluate the autonomous driving edge computing systems as we mentioned above.

CAVBench is a released benchmark suite specially developed for evaluating autonomous driving computing system performance [36]. It summarizes four application scenarios on CAVs: autonomous driving, real-time diagnostics, in-vehicle infotainment, and third-party applications, and chooses six classic and diverse real-world on vehicle applications as evaluation workloads, which are SLAM, object detection, object tracking, battery diagnostics, speech recognition, and edge video analysis. CAVBench takes four real-world data sets as the standard input to the six workloads and generates two categories of output metrics. One metric is an application perspective metric, which includes the execution time breakdown for each application, helping developers find the performance bottleneck in the application side. Another is a system perspective metric, which is the Quality of Service–Resource Utilization curve (QoS-RU curve). The QoS-RU curve is used to calculate the matching factor (MF) between the application and the computing platform on autonomous vehicles. The QoS-RU curve is considered as a quantitative performance index of the computing platform that helps researchers and developers optimize on-vehicle applications and CAVs computing architecture. We hope to see more research in the area of benchmarking for autonomous vehicle workloads, but currently, CAVBench serves as a good starting point to study edge computing systems for autonomous driving.

As autonomous driving is still a fast developing field, we hope to see continuous effort to incorporate more dynamic workloads and data to cover emerging autonomous driving usage scenarios. In addition, standardized scoring methods are required (but still missing) to rank different edge computing systems based on different optimization metrics.

B. Computing Architectures

Once we have standard benchmark suites, we start developing suitable architectures for autonomous driving workloads. Liu *et al.* [1] proposed a computer architecture for autonomous vehicles which fully utilizes hybrid heterogeneous hardware. In this work, the applications for autonomous driving are divided into three stages: sensing, perception, and decision-making. The authors compared the performance of different hardware running basic autonomous driving tasks and concluded that localization and perception as the bottlenecks of autonomous driving computing systems, and they also identified the need for different hardware accelerators for different workloads. Furthermore, the authors proposed and developed an autonomous driving computing architecture and software stack that is modular, secure, dynamic, high-performance, and energy-efficient. By fully utilizing heterogeneous computing components, such as CPU, GPU, and DSP, their prototype system on an ARM Mobile SoC consumes 11 W on average and is able to drive a mobile vehicle at 5 mph. In addition, the authors indicated that with more computing resources, the system would be able to process more data and would eventually satisfy the need of a production-level autonomous driving system.

Similarly, Lin *et al.* [37] explored the architectural constraints and acceleration of the autonomous driving system. The authors presented and formalized the design constraints of autonomous driving systems in performance, predictability, storage, thermal, and power when building autonomous driving systems. To investigate the design of the autonomous driving systems, the authors developed an end-to-end autonomous driving system based on machine learning algorithmic components. Through the experiments on this system, the authors identified three computational bottlenecks, namely, localization, object detection, and object tracking. To design a system which meets all the design constraints, the authors also explored three different accelerator platforms to accelerate these computational bottlenecks. The authors demonstrated that GPU, FPGA, and ASIC-accelerated systems could effectively reduce the tail latency of these algorithms. Based on these acceleration systems, the authors further explored the tradeoffs among performance, power, and scalability of the autonomous driving systems. Their conclusion is

that although power-hungry accelerators like GPUs predictably deliver the computation at low latency, their high power consumption, further magnified by the cooling load to meet the thermal constraints, significantly degrades the driving range and fuel efficiency of the vehicle. Finally, the authors indicated that computational capability remains the bottleneck that prevents them from benefiting from the higher system accuracy enabled by higher resolution cameras.

Interestingly, in their pioneering architectural exploration work, the authors discussed above both concluded that localization and perception are the computing bottlenecks and heterogeneous computing is a feasible approach to accelerate these workloads. For localization acceleration, Tang *et al.* [38] proposed a heterogeneous architecture for SLAM [38]. The authors first conducted a thorough study to understand visual inertial SLAM performance and energy consumption on existing heterogeneous SoCs. The initial findings indicate that existing SoC designs are not optimized for SLAM applications, and systematic optimizations are required in the IO interface, the memory subsystem, as well as computation acceleration. Based on these findings, the authors proposed a heterogeneous SoC architecture optimized for visual inertial SLAM applications. Instead of simply adding an accelerator, the authors systematically integrated direct IO, feature buffer, and a feature extraction accelerator. To prove the effectiveness of this design, the authors implemented the proposed architecture on a Xilinx Zynq UltraScale MPSoC and were able to deliver over 60 frames/s performance with average power less than 5 W. These results verify that the proposed architecture is capable of achieving performance and energy consumption optimization for visual inertial SLAM applications.

Similarly, to solve the localization computing problem, Zhang et al. [39] proposed an algorithm and hardware codesign methodology for visual–inertial odometry (VIO) systems, in which the robot estimates its ego-motion (and a landmark-based map) from the onboard camera and IMU data. The authors argued that scaling down VIO to miniaturized platforms (without sacrificing performance) requires a paradigm shift in the design of perception algorithms, and the authors advocated a codesign approach in which algorithmic and hardware design choices are tightly coupled. In detail, the authors characterized the design space by discussing how a relevant set of design choices affects the resource-performance tradeoff in VIO. Also, the authors demonstrated the result of the codesign process by providing a VIO implementation on specialized hardware and showing that such implementation has the same accuracy and speed of a desktop implementation while requiring a fraction of the power.

Besides academic research, PerceptIn has released a commercial production SLAM system titled DragonFly+ [40]. DragonFly+ is an FPGA-based real-time localization module with several advanced features: 1) hardware synchronizations among the four image channels as well as the IMU; 2) a direct IO architecture to reduce off-chip memory communication; and 3) a fully pipelined architecture to accelerate the image processing frontend. In addition, parallel and multiplexing processing techniques are employed to achieve a good balance between bandwidth and hardware resource consumption. Based on publicly available data, for processing four-way 720p images, DragonFly+ achieves 42 frames/s performance while consuming only 2.3 W of power. In comparisons, Nvidia Jetson TX1 GPU SoC achieves 9 frames/s at 7 W and Intel Core i7 achieves 15 frames/s at 80 W. Therefore, DragonFly+ is 3× more power efficient and delivers 5× of computing power compared to Nvidia TX1, and 34× more power efficient and delivers 3× of computing power compared Intel Core i7.

For perception acceleration, most recent research has focused on the acceleration of deep convolutional neural networks (CNNs). To enable CNN accelerators to support a wide variety of different applications with sufficient flexibility and efficiency, Liu *et al.* [41] proposed a novel domain-specific instruction set architecture (ISA) for neural network accelerators. The proposed ISA is a load-store architecture that integrates scalar, vector, matrix, logical, data transfer, and control instructions, based on a comprehensive analysis of existing neural network acceleration techniques. The authors demonstrated that the proposed ISA exhibits strong descriptive capacity over a broad range of neural network acceleration techniques and provides higher code density than general-purpose ISAs such as x86, Microprocessor without Interlocked Pipeline Stages (MIPS), and GPGPU.

Realizing that data movement is a key bottleneck for CNN computations, Chen et al. [42] presented a dataflow to minimize the energy consumption of data movement on a spatial architecture. The key is to reuse local data of filter weights and feature map pixels, or activations, in the high-dimensional convolutions, and minimize data movement of partial sum accumulations. The proposed dataflow adapts to different CNN shape configurations and reduces all types of data movement by maximally utilizing processing engine (PE) local storage, spatial parallelism, and direct inter-PE communication. Through the CNN configurations of AlexNet, evaluation experiments show that the proposed dataflow for more energy efficient than other dataflows for both convolutional and fully connected layers.

Also, memory access latency and throughput are often bottlenecks for neural network computations. One technique that has the potential to be used for the main memory is the metaloxide resistive random access memory (ReRAM). Moreover, with the crossbar array structure, ReRAM performs matrix–vector multiplication more efficiently and has been widely studied in accelerations of CNN applications. To accelerate CNN computations from the memory side, Chi *et al.* [43] proposed a processor-in-memory (PIM) architecture to accelerate CNN applications in ReRAM based main memory. In the proposed

design, a portion of ReRAM crossbar arrays is configured as accelerators for CNN applications or as normal memory for larger memory space. The microarchitecture and circuit designs are provided enable the morphable functions with an insignificant area overhead. Benefiting from both the PIM architecture and the efficiency of using ReRAM for CNN computation, the proposed design achieved significant performance improvement and energy savings, demonstrating the effectiveness of CNN accelerations from the memory side.

We have seen much recent progress on acceleration localization and perception functions, with many studies focused on reconfigurable fabrics [44]–[48]; however, with limited chip area, it is inefficient to integrate one accelerator for each perception and localization task. Therefore, the efficient utilization of reconfigurable fabrics is a major challenge for autonomous driving edge computing design. For instance, the object tracking task is triggered by the object recognition task and the traffic prediction task is triggered by the object tracking task. The data uploading task is also not needed all the time since uploading data in batches usually improves throughput and reduces bandwidth usage. To optimize for chip area and power consumption, one way is to have these tasks time-share an FPGA chip in the system. It has been demonstrated that using partial-reconfiguration techniques [49], an FPGA soft core could be changed within less than a few milliseconds, making time-sharing possible in real time.

In the near future, as more autonomous driving workloads and usage scenarios emerge, we look forward to the designs of more accelerators targeted for these workloads. Also, we expect to see more exploration studies on the cache, memory, and storage architectures for autonomous driving workloads. In addition, hardware security for autonomous driving is of utmost importance. Within a decade, the research community and the industry shall be able to come up with a "general-purposed" architecture design for autonomous driving workloads.

C. Runtime Systems

With heterogeneous architectures ready for autonomous driving tasks, the next challenge is how to dispatch incoming tasks to different computing units at runtime to achieve optimal energy efficiency and performance. This is achieved through a runtime layer. Designing runtime for heterogeneous autonomous driving systems is a whole new research area with tremendous potentials, as most existing runtime designs focus on either mapping one algorithm to one type of accelerators or on scheduling for homogeneous systems or heterogeneous systems with a single accelerator.

Several existing designs focus on mapping one deep learning or computer vision workload to heterogeneous architectures: Hegde *et al.* [50] propose a framework for easy mapping of CNN specifications to accelerators such as FPGAs, DSPs, GPUs, and Reduced Instruction Set Computer (RISC) multicores. Malik *et al.* [51] compare the performance and energy efficiency of computer vision algorithms on on-chip FPGA accelerators and GPU accelerators. Many studies have explored the optimization of deep learning algorithms on embedded GPU or FPGA accelerator [52], [53]. There have also been many projects on optimizing computer vision related tasks on embedded platforms. Honegger *et al.* [54] propose FPGA acceleration of embedded computer vision. Satria *et al.* [55] perform platform-specific optimizations of face detection on embedded GPU-based platform and reported real-time performance. Vasilyev *et al.* [56] evaluate computer vision algorithms on programmable architectures. Nardi *et al.* [57] present a benchmark suite to evaluate dense SLAM algorithms across desktop and embedded platforms in terms of accuracy, performance, and energy consumption. However, these designs did not consider the complexity of integrating the various kinds of workloads into a system and only focus on mapping one task to different accelerators.

Other existing designs focus on scheduling for heterogeneous architectures with one accelerator that has been broadly studied for single-ISA multiprocessors, such as asymmetric multicore architectures, i.e., big and small cores, and multi-ISA multiprocessors such as CPU with GPU. On the single-ISA multiprocessor side, much work has been done at the operating system level to map workload onto most appropriate core type in run time. Koufaty *et al.* [58] identified that the periods of core stalls is a good indicator to predict the core type best suited for an application. Based on the indicator, a biased schedule strategy was added to operating systems to improve system throughput. Saez *et al.* [59] proposed a scheduler that adds efficiency specialization and thread-level parallelism (TLP) specialization to operating systems to optimize throughput and power at the same time. Efficient specialization maps CPU-intensive workloads onto fast cores and memory-intensive workloads onto slow cores. TLP specialization uses fast cores to accelerate a sequential phase of parallel applications and use slow cores for the parallel phase to achieve energy efficiency. On the asymmetric multicore architectures side, Jiménez *et al.* [60] proposed a user-level scheduler for CPU with GPU like system. It evaluates and records the performance of a process on each PE at the initial phase. Then, based on this history information, it maps the application onto the best suited PE. Luk *et al.* [61] focus on improving the latency and energy consumption of a single process. It uses dynamic compilation to characterize workloads, determines optimal mapping, and generates codes for CPUs and GPUs.

Unlike existing runtime designs, Liu *et al.* [62] proposed PerceptIn Runtime (PIRT), the first runtime framework that is able to dynamically map various computer vision and deep learning workloads to multiple accelerators and to the cloud. The authors first conducted a comprehensive study of emerging robotic applications on heterogeneous SoC architectures. Based on the results, the authors designed and implemented PIRT to utilize not only the

on-chip heterogeneous computing resources but also the cloud to achieve high performance and energy efficiency. To verify its effectiveness, the authors have deployed PIRT on a production mobile robot to demonstrate that full robotic workloads, including autonomous navigation, obstacle detection, route planning, large map generation, and scene understanding, are efficiently executed simultaneously with 11 W of power consumption.

The runtime layer connects autonomous driving software and hardware, but there are several upcoming challenges in the design of runtime systems for autonomous driving. First, as the computing system becomes more heterogeneous, the runtime design becomes more complicated in order to dynamically dispatch incoming workloads. Second, as more edge clouds become available, the runtime system needs to be cloud-aware and able to dispatch workloads to edge clouds. Third, the runtime shall provide a good abstraction to hide all the low-level implementations.

D. Middleware

Robotic systems, such as autonomous vehicle systems, often involve multiple services, with a lot of dependencies in between. To facilitate the complex interactions between these services, to simplify software design, and to hide the complexity of low-level communication and the heterogeneity of the sensors, a middleware is required.

An early design of robotic middleware is Miro, a distributed object-oriented framework for mobile robot control, based on Common Object Request Broker Architecture technology (COBRA) [63]. The mirocore components have been developed under the aid of Adaptive Communications Environment, an object-oriented multiplatform framework for OS-independent interprocess, network, and real-time communication. Mirocore provides generic abstract services like localization or behavior engines, which is applied on different robot platforms. Miro supports several robotic platforms including Pioneers, B21, robot soccer robots, and various robotic sensors.

ORCA is an open-source component-based software engineering framework developed for mobile robotics with an associated repository of free, reusable components for building mobile robotic systems [64]. ORCA's project goals include enabling software reuse by defining a set of commonly-used interfaces; simplifying software reuse by providing libraries with a high-level convenient application program interface (API); and encouraging software reuse by maintaining a repository of components.

Urbi is open-source cross-platform software used to develop applications for robotics and complex systems [65]. Urbi is based on the UObject distributed C++ component architecture. Urbi includes the urbiscript orchestration language, a parallel and event-driven script language. In this design, UObject components are plugged into urbiscript as native objects to specify their interactions and data exchanges. UObjects are linked to the urbiscript interpreter, or executed as autonomous processes in "remote" mode, either in another thread, another process, a machine on the local network, or a machine on a distant network.

RT-middleware is a common platform standard for distributed object technology based robots based on distributed object technology [66]. RT-middleware supports the construction of various networked robotic systems through the integration of various network-enabled robotic elements called RT-components. In the RT-middleware, robotics elements, such as actuators, are regarded as RT-components, and the whole robotic system is constructed by connecting these RT-components. This distributed architecture helps developers to reuse the robotic elements and boosts the reliability of the robotic system.

OpenRDK is an open-source software framework for robotics for developing loosely coupled modules [67]. It provides transparent concurrency management, interprocess via sockets, and intraprocess via shared memory. Modules for connecting to simulators and generic robot drivers are provided.

The above-mentioned middleware projects mostly focused on providing a software component management framework for mobile robots and they were not used in autonomous vehicles. On the other hand, ROS has been widely used in autonomous vehicle development [21], mainly due to the popularity of ROS amount robotic developers and the richness of its software packages. However, as discussed in Section II, at its current state, ROS is not suitable for the production deployment of autonomous vehicles as it suffers from performance, reliability, and security issues.

PerceptIn Operating System (PIOS) is a developed extremely light-weighted middleware to facilitate the communications between services on production low-speed autonomous vehicles [68]. PIOS builds on top of Nanomsg, a networking library written in C that allows for easy integration of shared memory, transmission control protocol (TCP)/IP, in-process messaging, and web sockets while retaining efficiency [69]. Compared to ROS, PIOS is extremely lightweight, able to achieve a startup memory footprint of merely 10 KB, which is negligible comparing to 50 MB of ROS startup footprint.

OpenEI [70] is a lightweight software platform to equip edges with intelligent processing and data sharing capability. Due to the computing power limitation, weak data sharing, and collaborating, the deployment of artificial intelligence (AI)-based algorithms has faced many challenges. The goal of OpenEI is that any hardware, such as Raspberry Pi, will become an intelligent edge after deploying OpenEI.

The middleware layer facilitates communication between different autonomous driving services, but we summarize several challenges. First, the middleware should impose minimal computing overhead and memory footprint, thus making it scalable. Second, as some autonomous driving services may stay in edge clouds,

Table 1 Summary of Computing Edge Designs

Layer	Purpose	Proposed Solutions	Research Directions
Benchmark	Tools to evaluate edge computing systems	[36][26][27][28][29][30] [31][32][33][34][35]	usage scenarios; standardized, scoring methods to rank edge computing systems
Architecture	Hardware computing units to execute autonomous driving workloads	[1][37][38][39][40][41] [42][43]	workloads; cache and memory architecture design; non-volatile storage for critical data; hardware security
Runtime	Software layer to efficiently dispatch incoming tasks at run time to different computing units	[62][50][51][52][53][54] [55][56][58][59][60][61]	scheduler and dispatcher for highly heterogeneous computing systems; abstraction to hide low-level details; cloud awareness
Middleware	Software layer to enable complex interactions between autonomous driving services	[63][64][65][66][67][21] [68][70]	low overhead and memory footprint; edge-cloud interaction; security and reliability

the middleware should enable a smooth edge client and cloud communication. Third and most importantly, the middleware should be secure and reliable to guarantee the quality of service and autonomous vehicle safety.

E. Case Studies

To simultaneously enable multiple autonomous driving services, including localization, perception, and speech recognition workloads on affordable embedded systems, Tang *et al.* designed and implemented Π-Edge, a complete edge computing framework for autonomous robots and vehicles [68]. The challenge of designing such a system include the following: managing different autonomous driving services and their communications with minimal overheads, fully utilizing the heterogeneous computing resources on the edge device, and offloading some of the tasks to the cloud for energy efficiency.

To achieve these, first, the authors developed a runtime layer to fully utilize the heterogeneous computing resources of low-power edge computing systems; second, the authors developed an extremely lightweight middleware to manage multiple autonomous driving services and their communications; third, the authors developed an edge-cloud coordinator to dynamically offload tasks to the cloud to optimize client system energy consumption.

OpenVDAP is another real-world edge computing system which is a full-stack edge-based platform including vehicle computing unit, an isolation-supported and security and privacy-preserved vehicle operation system, an edge-aware application library, as well as task offloading and scheduling strategy [71]. OpenVDAP allows connected and autonomous vehicles (CAVs) to dynamically examine each task's status, computation cost, and the optimal scheduling method so that each service could be finished in near real time with low overhead. OpenVDAP is featured as a two-tier architecture via a series of systematic mechanisms that enable CAVs to dynamically detect service status and to identify the optimal offloading destination so that each service could be finished at the right time. In addition, OpenVDAP offers an open and free edge-aware library that contains how to access and deploy edge-computing-based vehicle applications and various common used AI models, thus enabling researchers and developers to deploy, test, and validate their applications in the real environment. Novel applications such as AutoVAPS benefit from it [72]. AutoVAPS is an Internet-of-Things (IoT)-enabled public safety service, which integrates body-worn cameras and other sensors on the vehicle for public safety.

HydraOne is an indoor experimental research and education platform for edge computing in CAVs scenarios [127]. HydraOne is a full-stack research and education platform from hardware to software, including mechanical components, vision sensors, as well as computing and communication system. All resources on HydraOne are managed by the Robot Operating System [76], and the data analytics on HydraOne is managed by OpenVDAP, which is a full-stack edge based platform [71]. HydraOne has three key characteristics: design modularization, resource extensibility, and openness, as well as function isolation, which allows users to conduct various research and education experiments of CAVs on HydraOne.

IV. INNOVATIONS ON THE V2X INFRASTRUCTURE

One effective method to alleviate the huge computing demand on autonomous driving edge computing systems is V2X technologies. V2X is defined as a vehicle communication system that consists of many types of communications: V2V, vehicle-to-network (V2N), vehicle-to-pedestrian (V2P), V2I, vehicle-to-device (V2D), and vehicle-to-grid (V2G). Currently, most research focuses on V2V and V2I. While conventional autonomous driving systems require costly sensors and edge computing equipment within the vehicle, V2X takes a different and potentially complimentary approach by investing in road infrastructure, thus alleviating the computing and sensing costs in vehicles.

With the rapid deployment of edge computing facilities in the infrastructure, more and more autonomous driving applications have started leveraging V2X communications to make the in-vehicle edge computing system more efficient. One popular direction is cooperative autonomous driving. The cooperation of autonomous driving edge computing system with V2X technology makes it possible to build a safe and efficient autonomous driving system [73]. However, the application and deployment of cooperative autonomous driving systems are still open research problems. In this section, we discuss the evolution of V2X technology and present four case studies of V2X

for autonomous driving: convoy driving, cooperative lane change, cooperative intersection management, and cooperative sensing.

A. Evolution of V2X Technology

In the development of V2X technology, many researchers have contributed solutions to specific challenges of V2X communication protocols. The intervehicle hazard warning (IVHW) system is one of the earliest studies to take the idea of improving vehicle safety based on communication [74]. The project is funded by the German Ministry of Education and Research and the French government. IVHW is a communication system in which warning messages are transmitted as broadcast messages in the frequency band of 869 MHz [75]. IVHW takes a local decision-making strategy. After the vehicle receives the message, it will do relevant checks to decide whether the warning message is relevant and should be shown to the driver. The majority of the research efforts have been made on the design of relevance check algorithms. However, as IVHW takes a broadcast mechanism to share the message, there is a huge waste in both bandwidth and computing resources.

Compared with the broadcast message in IVHW, *ad hoc* networking is a better solution to support multihop intervehicle communication [76]. FleetNet is another research project taking the idea of vehicle communication [77], and it is based on ad hoc networking. In addition, the FleetNet project also wants to provide a communication platform for Internet-protocol-based applications. FleetNet is implemented based on the IEEE 802.11 Wireless LAN system [78]. For V2V communication, if two vehicles are not directly connected wirelessly, it would need other vehicles to forward the message for them. Designing the routing and forwarding protocol is a major challenge. In order to meet the requirements for adaptability and scalability, FleetNet proposed a position-based forwarding mechanism. The idea is to choose the next hop to forward the message based on the geographical location of the vehicle.

CarTALK 2000 is also a project working on using an ad hoc communication network to support co-operative driver assistance applications [79]. There is a major challenge for ad hoc-based routing in V2V communication because the vehicle network topology is dynamic and the number of vehicles is frequently changing [80]. In order to solve the problem, a spatial aware routing algorithm is proposed in CarTALK 2000, which takes spatial information like underlying road topology into consideration. Compared with FleetNet, CarTALK 2000 achieves better performance as it uses spatial information as additional input for routing algorithm. Another similar part of CarTALK 2000 and FleetNet is that they are both based on WLAN technology. AKTIV is another project that is the first one tried to apply cellular systems in driving safety applications [81]. One of the reasons that FleetNet and CarTALK 2000 project built their system based on WLAN technology is that the latency of safety-related applications required is less than 500 ms. However, with the assumption that a long-term evolution (LTE) communication system is greatly further developed, cellular systems are a good choice for sparse vehicle networking.

Meanwhile, several research projects have focused on warning applications based on V2V communication. Wireless local danger warning (WILLWARN) proposed a risk detection approach based on in-vehicle data. The warning message includes obstacles, road conditions, low visibility, and construction sites [82]. Unlike other projects focusing on the V2X technology itself, WILLWARN focuses on enabling V2X technology in some specific scenario such as the danger spot. Suppose some potential danger is detected in a specific location, but there is no vehicle within the communication range that supports the V2X communication technology to share the warning message [83]. To share warning messages, WILLWARN proposed a decentralized distribution algorithm to transmit the warning message to vehicles approaching the danger spot through V2V communication. The project Network on Wheels (NoW) was one work which takes the idea of FleetNet to build vehicle communication based on 802.11 WLAN and *ad hoc* networking [84]. The goal of NoW is to set up a communication platform to support both mobility and Internet applications. For example, a hybrid forwarding scheme considering both network layer and application layer is developed. Also, security and scalability issues are discussed in NoW.

As the infrastructure also plays a very important part in V2X technology, several efforts focus on building safety applications based on the cooperation with infrastructure. SAFESPOT is an integrated project that is aimed at using roadside infrastructure to improve driving safety [85]. Through combining information from the on-vehicle sensors and infrastructure sensors, SAFESPOT detects dangerous situations and shares the warning messages in real time. Also, the warning forecast is improved from the milliseconds level to the seconds level, thus giving the driver more time to prepare and take action. Five applications are discussed in SAFESPOT, including hazard and incident warning, speed alert, road departure prevention, cooperative intersection collision prevention, and safety margin for assistance and emergency vehicles [86].

In 2007, a nonprofit organization called the Car 2 Car Communication Consortium (C2C-CC) was set up to combine all solutions from different former projects to make a standard for V2X technology. Since 2010, the focus of work on V2X technology has moved from research topics to the real environment deployment of the whole intelligent transportation system (ITS). One of the most popular deploy projects is sim^{TD} [73], targeted on testing the V2X applications in a real metropolitan field. In sim^{TD}, all vehicles connect with each other through dedicated short-range communications (DSRC) technology that is based on IEEE 802.11p. Meanwhile, vehicles also communicate with roadside infrastructure using IEEE 802.11p. The system

Table 2 Evolution of V2X Communication Technology

Research	Application Scenario	Proposed Solutions	Communication Protocol
IVHW	Safe driving	Warning message are transmitted as broadcast message, and vehicle takes a local decision-making strategy.	Frequency band of 869MHz
FleetNet	Safe driving, internet protocol based applications	Uses ad-hoc networking to support multi-hop inter-vehicle communications, proposes a position-based forwarding mechanism	IEEE 802.11 Wireless LAN
CarTALK 2000	Cooperative driver assistance applications	Uses ad-hoc communication network to support co-operative driver assistance applications, a spatial aware routing algorithm which takes some spatial information like underlying road topology into consideration to solve	IEEE 802.11 Wireless LAN
AKTIV	Safe driving	WLAN technology is that the latency of safety related applications required to be less than 500ms	Cellular systems
WILLWARN	Warning applications	Propose a risk detection approach based on in-vehicle data. The warning message includes obstacles, road conditions, low visibility, and construction sites. A decentralized distribution algorithm to transmit the warning message to vehicles approaching the danger spot through V2V communication	IEEE 802.11 Wireless LAN
NoW	Mobility and internet applications	A hybrid forwarding scheme considering both network-layer and application-layer is developed. Also, some security and scalability issues are discussed.	IEEE 802.11 Wireless LAN
SAFESPOT	Safe driving	SAFESPOT is an integrated project which aims at using roadside infrastructure to improve driving safety. detects dangerous situations and shares the warning messages in real time	IEEE 802.11 Wireless LAN
sim^{TD}	Traffic Manipulation, safe driving, and internet based applications	Real environment deployment of the whole Intelligent Transportation System. The system architecture of simTD can be divided into three parts: ITS vehicle station, ITS roadside station, and ITS central station.	IEEE 802.11p (DSRC)

architecture of sim^{TD} is divided into three parts: the ITS vehicle station, the ITS roadside station, and the ITS central station. Applications for testing in sim^{TD} include traffic situation monitoring, traffic flow information and navigation, traffic management, driving assistance, local danger alert, and Internet-based applications.

Cellular V2X (C-V2X) is designed as a unified connectivity platform which provides low-latency V2V and V2I communications [87]. It consists of two modes of communications. The first mode uses direct communication links between vehicles, infrastructure, and pedestrian. The second mode relies on network communication, which leverages cellular networks to enable vehicles to receive information from the Internet. C-V2X further extends the communication range of the vehicle and it supports the higher capacity of data for information transmission for vehicles.

B. Cooperative Autonomous Driving

Cooperative autonomous driving is divided into two categories: one is cooperative sensing and the other is cooperative decision [88]. Cooperative sensing focuses on sharing sensing information between V2V and V2I. This data sharing increases the sensing range of autonomous vehicles, making the system more robust. The cooperative decision enables a group of autonomous vehicles to cooperate and make decisions.

Several studies have focused on the exploration of applications for cooperative autonomous driving. In [88], four use cases including convoy driving, cooperative lane change, cooperative intersection management, and cooperative sensing are demonstrated. According to the design of AutoNet2030 [89], a convoy is formed of vehicles on multilanes into a group and the control of the whole group is decentralized. The safety and efficient control of the convoy require high-frequency exchanges of each vehicle's dynamic data. As shown in Fig. 4, a roadside edge server and a cloud server are used to coordinate and manage the vehicles and convoys to go through crossroads safely. One convoy control algorithm in [90] only exchanges dynamic information of the nearby vehicle rather than for all the vehicles within a convoy. This design makes the algorithm easy to converge.

Cooperative lane change is designed to make vehicles or convoys to collaborate when changing lanes. Proper cooperative lane change cannot only avoid traffic accidents but it also reduces traffic congestion [91]. MOBIL [92] is a general model whose objective is to minimize overall braking induced by lane changes.

Cooperative intersection is also helpful for safe driving and traffic control. The World's Smartest Intersection in Detroit [93] focuses on safety and generates data that

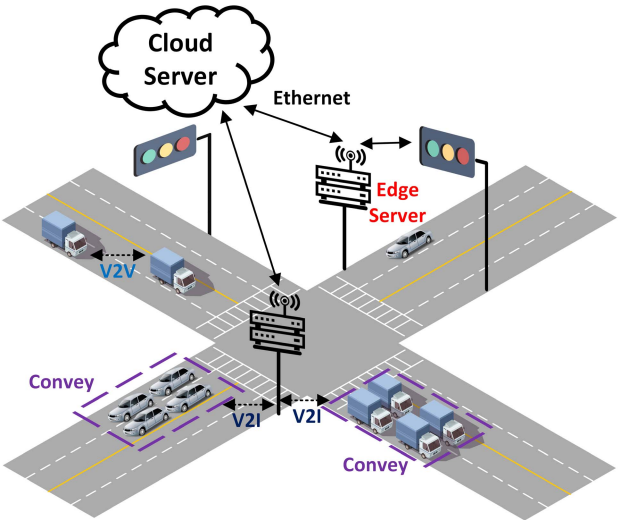

Fig. 4. *V2X communications in crossroads.*

pinpoint areas where traffic-related fatalities and injuries are reduced. Effective cooperative intersection management is based on the coordination mechanism between V2V and V2I.

Cooperative sensing increases the autonomous vehicle sensing range through V2X communication. Meanwhile, cooperative sensing also helps in cutting the cost of building autonomous driving. As vehicles rely more on the sensors deployed on roadside infrastructure, the cost of on-vehicle sensors is reduced. In the future, sensor information may become a service to the vehicle provided by the roadside infrastructure.

V2X networking infrastructure is also a very important aspect of cooperative autonomous driving. Heterogeneous vehicular network (HetVNET) [94] is an initial work on networking infrastructure to meet the communication requirements of the ITS. HetVNET integrates LTE with DSRC [95] because relying on the single wireless access network cannot provide satisfactory services in dynamic circumstances. In [96], an improved protocol stack is proposed to support multiple application scenarios of autonomous driving in HetVNET. In the protocol, the authors redefined the control messages in HetVNET to support autonomous driving.

Similarly, the vehicular delay-tolerant network (VDTN) [97] is an innovative communication architecture, which is designed for scenarios with long delays and sporadic connections. The idea is to allow messages to be forwarded in short-range WiFi connections and reach the destination asynchronously. This property enables VDTN to support services and applications even when there is no end to end path in current Vehicular Ad hoc Network (VANET). Dias *et al.* [98] discuss several cooperation strategies for VDTN. The challenge for cooperation in VDTN is how to coordinate the vehicle node to share their constrained bandwidth, energy resources, and storage with one another. Furthermore, an incentive mechanism that rewards or punishes vehicles for cooperative behavior is proposed.

In order to support seamless V2X communication, handover is also a very important topic for V2X networking infrastructure. Due to the dynamic changing of the networking topology and the relatively small range of the communication coverage, the handover mechanism in a cellular network is no longer suitable for VANET. Based on proactive resource allocation techniques, Ghosh *et al.* [99] propose a new handover model for VANET. With the help of proactive handover, cooperative services are migrated through RSUs with the moving of the vehicle. Hence, proper designing of proactive handover and resource allocation is essential for developing reliable and efficient cooperative systems.

The development of edge computing in the automotive industry is also very inspiring. Automotive Edge Computing Consortium (AECC) is a group formed by automotive companies to promote edge computing technologies in future automobiles [100]. According to an AECC white paper from 2018, the service scenarios include intelligent driving, HD map, V2Cloud cruise assist, and several extended services like finance and insurance. In addition, the white paper discusses the service requirements in terms of data source, the volume of data generated in the vehicle, target data traffic rate, response time, and required availability.

C. Challenges

In order to guarantee the robustness and safety of autonomous driving systems, autonomous vehicles are typically equipped with numerous expensive sensors and computing systems, leading to extremely high costs and preventing ubiquitous deployment of autonomous vehicles. Hence, V2X is a viable solution in decreasing the costs of autonomous driving vehicles as V2X enables information sharing between vehicles and computation offloading to RSUs. There are several challenges in achieving cooperative autonomous driving. Here, we discuss the challenges and our vision for application scenario of cooperative decision and cooperative sensing.

1) Cooperative Decision: The challenge of cooperative decisions is handling the dynamic changing topology with a short-range coverage of V2X communications. The design of VDTN is a good hint to solve this challenge. Effective proactive handover and resource allocation are potential solutions. Also, the coming 5G wireless communication [101] also provides a way to handle this challenge.

2) Cooperative Sensing: The main challenge of cooperative sensing is sharing the information from infrastructure sensors to autonomous vehicles in real time, and the other challenge is to dynamically trade off the cost of infrastructure sensors and on-vehicle sensors. For the first challenge, the promising edge computing technology is used to solve the problem [71] because edge computing enables the edge node(vehicle) and edge server(infrastructure) to conduct computation and compression to provide real-time performance. In addition, the tradeoff of cost on infrastructure sensors and on-vehicle sensors will be determined by the automobile market. Both the government and companies will invest much money to support edge intelligence.

V. VEHICULAR EDGE SECURITY

The previous sections reviewed innovations in an edge computing infrastructure to make autonomous driving computing more efficient in terms of performance and energy consumption. As mentioned above, each autonomous vehicle is equipped with or supported by dozens of computing units in the edge and cloud to process the sensor data, to monitor the vehicles' status and to control the mechanical components. Hence, the security threats against these computing units are of paramount concern. Specifically, the attacks targeting autonomous

vehicles could cause terrible traffic accidents, threatening both personal and public safety. In this Section, we review recent advancements in the security of autonomous vehicles, including sensor security, operating system security, control system security, and communication security. These security problems cover different layers of the autonomous driving edge computing stack.

A. Sensors Security

The autonomous vehicles are equipped with various sensors (camera, GNSS, LiDAR, etc.) to enable the perception of the surrounding environments. The most direct security threats against autonomous vehicles are attacks against the sensors. With this attack method, attackers generate incorrect messages or completely block sensor data so as to interfere in the autonomous driving without hacking into the computing system. According to the working principle of sensors, the attackers have many specific attack methods to interfere, blind, or spoof each of them [102].

A camera is the basic visual sensor in autonomous driving systems. Modern autonomous vehicles are usually equipped with multiple cameras with the same or different lenses [1], [26]. In general, many autonomous driving perception workloads take camera images as inputs, for example, object detection and object tracking. The attackers place fake traffic lights, traffic signs, and traffic objects (cars or pedestrians) to spoof autonomous vehicles and let them make the wrong decisions [103]. The cameras are also interfered with infrared so the attackers use high-brightness IR laser to blind the cameras, thus preventing these cameras from providing effective images for the perception stage [103], [104].

Autonomous vehicles use GNSS and inertial navigation system (INS) sensors to update vehicles' real-time locations. Typical attacks against GNSS sensors are jamming and spoofing. The attackers could use out-of-band or in-band signals to intentionally interfere with the function of the GNSS receiver [105]. They also deploy GNSS transmitter near the autonomous vehicles to deceive the GNSS receiver by replicating original signals and providing false locations [103], [105]. In addition, the INS sensors are sensitive to magnetic fields, so an extra and powerful magnetic field could effectively interfere with the INS sensors to produce incorrect orientation of the vehicles under attack.

LiDAR provides point clouds of the vehicle's surroundings to generate 3-D sensing data of the environments. LiDAR measures distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses. A smart surface that is absorbent or reflective deceives LiDAR sensors to miss real obstacles in traffic [102], and light laser pulse illuminating the LiDAR could also manipulate the data sensed by the LiDAR, deceiving the LiDAR to sense objects in incorrect positions and distances [103]. For ultrasonic sensors and radars, which are mostly used for passive perception and the last line of defense for the autonomous vehicles, Yan *et al.* [106] have successfully spoofed and jammed these two kinds of sensors in the Tesla Autopilot system via the specific signal generator and transmitter.

B. Operating Systems Security

One widely used autonomous vehicle operating system is ROS. Attackers target ROS nodes and/or ROS messages. In the ROS running environment, there is no authentication procedure for message passing and new node creation. The attackers use the IP addresses and ports on the master node to create a new ROS node or hijack an existing one without further authentication [107]. If service on the node keeps consuming system resources, for example, memory footprint or CPU utilization, it will impact the performance of other normal ROS nodes, even crashing the whole autonomous driving system. The attackers also use the controlled ROS node to send manipulated messages to disturb other nodes running and output.

For the attacks on ROS messages, the first security threat is message capture. The attackers monitor and record every ROS message topics via the IP address and port on the master node. The recorded data are stored in the ROS bag file, and the attackers play the ROS bag file to resend history ROS messages, which will affect current ROS messages communication [107]. The message passing mechanism of ROS is based on the socket communication so the attackers sniff the network packets to monitor and intercept the ROS messages remotely without hacking in the master node [108], [109]. The attacks on ROS messages do not need to start or hijack a ROS node, but the security threat level is not lower than the attack with the ROS node method.

C. Control Systems Security

In modern vehicles, many digital devices and mechanical components are controlled by electronic control units (ECUs). The ECUs are connected to each other via the digital buses, which form the in-vehicle network. Controller area network (CAN) is the primary bus protocol in the vehicle [110]. CAN is the typical bus topology; there is no master/slave node concept in the CAN bus so any node connected to the CAN bus can send a message to any other node. Thus, the CAN network usually uses the priority to control accesses to the bus. The CAN network is isolated to the external network, but the attackers hack the digital devices in the vehicle to attack the CAN and ECUs indirectly, which is very dangerous to the vehicle and the public.

There are many attack surfaces of the CAN bus. First is the onboard diagnostics (OBD)-II port, which is used for vehicle status diagnostics, ECU firmware update, and even vehicle control. The OBD-II port is connected to the CAN bus, so the attackers use the OBD-II device and diagnostic software to sniff the message on the bus or control the vehicle [111], [112]. The attackers easily gain access to the CAN bus through OBD-II ports. Second is the media player (e.g., CD player) in the vehicle. The media player

Table 3 Security Threats and Defense Technologies of CAVs

Security Category	Security Threats	Defense Technologies
Sensors	Spoofing cameras by fake traffic objects.	**Multi-sensor data fusion**: System check and correct the sensor data from multiple sources.
	Jamming GPS receiver by high-power false GPS transmitter.	
	Jamming IMU sensor by powerful magnetic field.	
	Jamming LiDAR by light laser pulse.	
	Jamming and Spoofing ultrasonic sensors and MMW radars by specific signal generator.	
Operating Systems	Hijacking ROS node to consume system resources.	**Linux container**: Use the container technology to throttle the resource utilization of each ROS node. **Trusted execution environment**: Run the key ROS node in a trusted execution environment.
	Hijacking ROS node to send manipulated messages.	
	Sniffing ROS message to steal private data.	
	Repeating the intercepted ROS message to disturb other ROS nodes.	
Control Systems	Hijacking CAN bus by OBD-II port.	**Message encryption**: Encrypt message in CAN bus.
	Hijacking CAN bus by media player.	
	Hijacking CAN bus by Bluetooth.	
	Injecting manipulated messages on CAN bus.	
	DoS attack on CAN bus.	
V2X	DoS and DDoS attack on vehicle and infrastructure.	**Authentication and certification**: The node access the V2X network should be authenticated and provide security certificates and keys.
	Sybil attack by creating multiple fake vehicles in road.	
	Sniffing private data by short-range wireless protocol.	
	Broadcasting fake traffic information to nearby vehicles.	

needs to receive the control message from the driver and send the status to the screen (UI), so the media player usually has a connection to the CAN. The attackers easily flash the malicious code to a CD; when the driver plays the CD, the malicious code attacks the CAN bus [113]. In addition, the attackers utilize the Bluetooth interface in the vehicle. Modern vehicles support Bluetooth connections to smartphones. The attackers use smartphones to upload malicious applications via Bluetooth to take over the CAN bus, or they sniff the vehicle status via Bluetooth. It is important to note that the attackers use this interface to attack the vehicle remotely.

Once the attacker hijacks the CAN bus, there are security threats to the CAN network [111]. First is broadcast trouble. The CAN message is broadcast to all nodes, and the attackers capture and reverse-engineer these messages and inject new messages to induce various actions. Second is a denial-of-service (DoS) attack. The CAN protocol is extremely vulnerable to the DoS attack because of the limited bandwidth. In addition to message flooding attacks, if one hijacked node keeps claiming highest priority in the network, it will cause all other CAN nodes to back off, and the whole CAN network will crash. Last is no authentication fields. The CAN message does not contain the authentication fields, which means that any node sends a packet to any other node without an authentication process so the attackers use this to control any node in the CAN network.

D. V2X Security

With V2X, vehicles access the Internet to obtain real-time traffic data (e.g., real-time map and weather data) or leverage the cloud computing for autonomous driving [23], and the vehicle also communicates with other nodes in the V2X network via some emerging technologies (e.g., DSRC and LET-V) [114]. This V2X network creates many new application scenarios for the CAV, but it also brings more security problems to the vehicles [115]–[117].

The traditional Internet and *ad hoc* networks suffer from many security threats, which may occur in the V2X network but with different manifestations. DoS and distributed DoS (DDoS) attacks are two basic attack methods on the Internet. In V2X networks, every node is an attacker or a victim, causing various traffic problems [118]. If the infrastructure is the victim, it cannot provide real-time service for the nearby vehicles. In contrast, if the vehicle is the victim, it cannot receive the messages from the infrastructure or cloud, and the DoS attack also interferes with the performance of other tasks on the vehicle, causing unacceptable latency of some autonomous driving applications [102].

In V2X networks, the attackers create multiple vehicles on the road with the same identity or remain anonymous, which we call the Sybil attack [119]. The Sybil attack may enforce the vehicles running on the road to make way for the fake vehicles and prevent other vehicles from driving on this road because they are deceived to think there is a traffic jam. Information forgery is also a common attack; a vehicle changes its identity or sends fabricated messages to V2X networks, thus preventing itself from being detected or to shirk their responsibilities [120]. There are many other traditional network threats, such as a replay attack and a block hole attack, but the attack method is similar to the threats mentioned above.

The V2X network brings new types of network nodes, such as the infrastructure and pedestrian so it will have new threats that are rare on the traditional Internet. The first one is about privacy. The communication between V2P and V2I may be based on some short-range protocol (Bluetooth low energy (BLE) and DSRC); if access authentication is not strict, privacy of drivers and pedestrians will be exposed [121]. The second one is about infrastructure. If infrastructure (RSU) has been attacked and fake traffic information is broadcast, it influences the running state of the nearby vehicle.

E. Security for Edge Network and Platforms

Security is a critical topic for edge computing, so several works on security for the edge computing system in some general scenarios may provide solutions for security

problems in CAV scenarios. The related work is divided into two categories: network in edge and running environment for edge computing.

Bhardwaj *et al.* [122] proposed ShadowNet, which deploys the edge functions on the distributed edge infrastructure and aggregates that information about the IoT traffic to detect an imminent IoT-DDoS. ShadowNet detects IoT-DDoS ten times faster than existing approaches and also prevents 82% of traffic to enter the Internet infrastructure, reducing security threats. Yi *et al.* [123] summarized the method that uses the software-defined network (SDN) to solve edge network security problems, such as network monitoring and intrusion detection and network resource access control. This kind of work will help us solve the related network threats in CAV scenarios.

Ning *et al.* [124] evaluated several trusted execution environments (TEEs) on heterogeneous edge platforms, such as Intel SGX, ARM TrustZone, and AMD SEV, and deployed the TEEs on edge computing platform to efficiently improve the security with a low-performance overhead [124]. Kernel level resource auditing tool (KLRA) [125] is a KLRA for IoT and edge operating system security. KLRA takes fine-grained events measured with low cost and reports the relevant security warning the first time when the behavior of the system is abnormal with this device. This kind of work will help us solve the security problems in the operating system on CAVs.

VI. CONCLUSION

Safety is the most important requirement for autonomous vehicles; hence, the challenge of designing an edge computing ecosystem for autonomous vehicles is to deliver enough computing power, redundancy, and security to guarantee the safety of autonomous vehicles. In this paper, we reviewed recent advancements and presented challenges in building edge computing systems for autonomous vehicles, mainly in three categories: edge computing system designs, V2X applications, and autonomous vehicle security.

In edge computing system designs, we face a systematic challenge for delivering more computing power with reasonable energy consumption, to guarantee the safety of autonomous vehicles, even at high speed. First, as more autonomous driving edge computing workloads emerge, we need to continue developing standard benchmarks to more accurately evaluate edge computing systems. Second, although we have seen some recent progress in utilizing heterogeneous computing architectures for autonomous driving workloads, more research is required to improve the edge computer architecture for autonomous driving. The aim is to deliver more computing power with a limited chip area and energy budget. Third, with the evolution of computer architectures, we need to design runtime layers for efficient mapping of the incoming dynamic workloads. Fourth, a reliable, robust, and production-quality middleware is still missing to bind various autonomous driving services together. For large-scale deployment of autonomous vehicles, we hope to see more production-quality full-edge computing systems that incorporate innovations in these layers.

V2X enables information sharing between vehicles and computation offloading to RSUs. Hence, V2X not only alleviates the stress on edge computing systems but also provides redundancy for sensing, perception, and decision tasks. Therefore, we see V2X as an effective way to improve the safety of autonomous vehicles. Yet, several technical challenges remain such as how the edge computing systems on different vehicles cooperate to make sensing and planning decisions, and how to share the information from infrastructure sensors to autonomous vehicles in real time, as well as to dynamically trade off the cost on infrastructure sensors and on-vehicle sensors. With more research exploring V2X technologies, we hope to see more industrial standards emerging to define how vehicles cooperate with each other and with the infrastructure.

The safety of autonomous vehicles is at risk if security is compromised at any level. As each autonomous vehicle is equipped with numerous sensors and computing units, an attacker targets one of the sensors, the computing systems, the control systems, or the communication networks to confuse, blind, or even take over control of the vehicle under attack, leading to catastrophic accidents. Hence, the challenge is to guarantee security at all levels of the stack. Although in recent years there have been some interesting proposals regarding protecting the security of autonomous vehicles, such as sensor fusion, more research is required before we deploy autonomous vehicles on a large scale. We hope to see more research on defining attack surfaces against autonomous driving edge computing ecosystems and proposals to protect autonomous vehicles from these attacks. Most importantly, we urgently need industrial security verification standards to certify whether an autonomous vehicle is secure enough to run on public roads. ■

REFERENCES

[1] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[2] S. Liu, J. Peng, and J.-L. Gaudiot, "Computer, drive my car!" *Computer*, vol. 50, no. 1, p. 8, 2017.

[3] S. Liu, L. Li, J. Tang, S. Wu, and J.-L. Gaudiot, "Creating autonomous vehicle systems," *Synth. Lect. Comput. Sci.*, vol. 6, no. 1, p. 186, 2017.

[4] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended Kalman filter based SLAM," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1036–1049, Oct. 2007.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[6] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: Science and Systems*, vol. 4. 2007, p. 1.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[8] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 850–865.

[9] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2008, pp. 1149–1154.

[10] S. M. LaValle, *Planning Algorithms*. Cambridge,

U.K.: Cambridge Univ. Press, 2006.
[11] J. V. Frasch *et al.*, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2013, pp. 4136–4141.
[12] *Nvidia Drive PX*. Accessed: Dec. 28, 2018. [Online]. Available: https://www.nvidia.com/en-au/self-driving-cars/drive-px/
[13] Texas Instruments TDA. Accessed: Dec. 28, 2018. [Online]. Available: http://www.ti.com/processors/automotive-processors/tdax-adas-socs/overview.html
[14] *CEVA-XM4*. Accessed: Dec. 28, 2018. [Online]. Available: https://www.ceva-dsp.com/product/ceva-xm4/
[15] Intel Cyclone. Accessed: Dec. 28, 2018. [Online]. Available: https://www.intel.com/content/www/us/en/fpga/devices.html
[16] Xilinx Zynq. Accessed: Dec. 28, 2018. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
[17] Mobileye Eyeq. Accessed: Dec. 28, 2018. [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/
[18] E. D. Jensen, C. D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems," in *Proc. RTSS*, vol. 85, 1985, pp. 112–122.
[19] D. Hildebrand, "An architectural overview of QNX," in *Proc. USENIX Workshop Microkernels Other Kernel Archit.*, 1992, pp. 113–126.
[20] Vxworks. Accessed: Dec. 28, 2018. [Online]. Available: https://www.windriver.com/products/vxworks/
[21] M. Quigley *et al.*, "ROS: An open-source Robot Operating System," in *Proc. ICRA Workshop Open Sour. Softw.*, Kobe, Japan, 2009, vol. 3. no. 2, p. 5.
[22] *ISO 26262*. Accessed: Dec. 28, 2018. [Online]. Available: https://www.iso.org/standard/43464.html
[23] S. Liu, J. Tang, C. Wang, Q. Wang, and J.-L. Gaudiot, "A unified cloud platform for autonomous driving," *Computer*, vol. 50, no. 12, pp. 42–49, 2017.
[24] J. Tang, S. Liu, C. Wang, and C. Liu, "Distributed simulation platform for autonomous driving," in *Proc. Int. Conf. Internet Vehicles* Springer, 2017, pp. 190–200.
[25] J. Jiao, "Machine learning assisted high-definition map creation," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 367–373.
[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.
[27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 573–580.
[29] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond PASCAL: A benchmark for 3D object detection in the wild," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2014, pp. 75–82.
[30] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. (2015). "MOTChallenge 2015: Towards a benchmark for multi-target tracking." [Online]. Available: https://arxiv.org/abs/1504.01942
[31] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. (2016). "MOT16: A benchmark for multi-object tracking." [Online]. Available: https://arxiv.org/abs/1603.00831
[32] S. K. Venkata *et al.*, "SD-VBS: The San Diego vision benchmark suite," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2009, pp. 55–64.
[33] J. Clemons, H. Zhu, S. Savarese, and T. Austin, "MEVBench: A mobile computer vision benchmarking suite," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Nov. 2011, pp. 91–102.
[34] L. Nardi *et al.*, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 5783–5790.
[35] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2011, pp. 127–136.
[36] Y. Wang, S. Liu, X. Wu, and W. Shi, "CAVBench: A benchmark suite for connected and autonomous vehicles," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 30–42.
[37] S.-C. Lin *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Architectural Support Program. Lang. Oper. Syst.*, 2018, pp. 751–766.
[38] J. Tang, B. Yu, S. Liu, Z. Zhang, W. Fang, and Y. Zhang, "π-SOC heterogeneous SOC architecture for visual inertial SLAM applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–6.
[39] Z. Zhang, A. A. Suleiman, L. Carlone, V. Sze, and S. Karaman, "Visual-inertial odometry on chip: An algorithm-and-hardware co-design approach," Tech. Rep., 2017.
[40] W. Fang, Y. Zhang, B. Yu, and S. Liu, "DragonFly+: FPGA-based quad-camera visual SLAM system for autonomous vehicles," in *Proc. IEEE HotChips*, 2018, p. 1.
[41] S. Liu *et al.*, "Cambricon: An instruction set architecture for neural networks," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 393–405, 2016.
[42] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 367–379, 2016.
[43] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, Jun. 2016.
[44] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "DLAU: A scalable deep learning accelerator unit on FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 3, pp. 513–517, Mar. 2017.
[45] O. Rahnama, D. Frost, O. Miksik, and P. H. Torr, "Real-time dense stereo matching with ELAS on FPGA-accelerated embedded devices," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2008–2015, Jul. 2018.
[46] W. Fang, Y. Zhang, B. Yu, and S. Liu, "FPGA-based ORB feature extraction for real-time visual SLAM," in *Proc. Int. Conf. Field Program. Technol. (ICFPT)*, Dec. 2017, pp. 275–278.
[47] X. Wei *et al.*, "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *Proc. 54th Annu. Design Autom. Conf.*, 2017, p. 29.
[48] Y. Guan *et al.*, "FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates," in *Proc. IEEE 25th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2017, pp. 152–159.
[49] S. Liu, R. N. Pittman, A. Forin, and J.-L. Gaudiot, "Achieving energy efficiency through runtime partial reconfiguration on reconfigurable systems," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 3, p. 72, 2013.
[50] G. Hegde *et al.*, "CaffePresso: An optimized library for deep learning on embedded accelerator-based platforms," in *Proc. CASES*, 2016, p. 14.
[51] M. Malik *et al.*, "Architecture exploration for energy-efficient embedded vision applications: From general purpose processor to domain specific accelerator," in *Proc. ISVLSI*, Jul. 2016, pp. 559–564.
[52] L. Cavigelli, M. Magno, and L. Benini, "Accelerating real-time embedded scene labeling with convolutional networks," in *Proc. DAC*, 2015, p. 108.
[53] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. FPGA*, 2016, pp. 26–35.
[54] D. Honegger *et al.*, "Real-time and low latency embedded computer vision hardware based on a combination of FPGA and mobile CPU," in *Proc. IROS*, Sep. 2014, pp. 4930–4935.
[55] M. T. Satria *et al.*, "Real-time system-level implementation of a telepresence robot using an embedded GPU platform," in *Proc. DATE*, Mar. 2016, pp. 1445–1448.
[56] A. Vasilyev, N. Bhagdikar, A. Pedram, S. Richardson, S. Kvatinsky, and M. Horowitz, "Evaluating programmable architectures for imaging and vision applications," in *Proc. MICRO*, 2016, pp. 38–49.
[57] L. Nardi *et al.*, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *Proc. ICRA*, May 2015, pp. 5783–5790.
[58] D. Koufaty, D. Reddy, and S. Hahn, "Bias scheduling in heterogeneous multi-core architectures," in *Proc. 5th Eur. Conf. Comput. Syst.*, 2010, pp. 125–138.
[59] J. C. Saez, M. Prieto, A. Fedorova, and S. Blagodurov, "A comprehensive scheduler for asymmetric multicore systems," in *Proc. 5th Eur. Conf. Comput. Syst.*, 2010, pp. 139–152.
[60] V. J. Jiménez, L. Vilanova, I. Gelado, M. Gil, G. Fursin, and N. Navarro, "Predictive runtime code scheduling for heterogeneous architectures," in *Proc. HiPEAC*, vol. 9, 2009, pp. 19–33.
[61] C.-K. Luk, S. Hong, and H. Kim, "Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 45–55.
[62] L. Liu, S. Liu, Z. Zhang, B. Yu, J. Tang, and Y. Xie. (2018). "Pirt: A runtime framework to enable energy-efficient real-time robotic applications on heterogeneous architectures." [Online]. Available: https://arxiv.org/abs/1802.08359
[63] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro-middleware for mobile robot applications," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 493–497, Aug. 2002.
[64] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback, "Towards component-based robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Aug. 2005, pp. 163–168.
[65] J.-C. Baillie, "Urbi: Towards a universal robotic low-level programming language," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Aug. 2005, pp. 820–825.
[66] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, "RT-middleware: Distributed component middleware for RT (robot technology)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Aug. 2005, pp. 3933–3938.
[67] D. Calisi, A. Censi, L. Iocchi, and D. Nardi, "OpenRDK: A modular framework for robotic software development," in *Proc. IROS*, Sep. 2008, pp. 1872–1877.
[68] J. Tang, S. Liu, B. Yu, and W. Shi. (2018). "PI-Edge: A low-power edge computing system for real-time autonomous driving services." [Online]. Available: https://arxiv.org/abs/1901.04978
[69] M. Sustrik. (2016). *Nanomsg*. [Online]. Available: https://www.nanomsg.org
[70] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, and W. Shi, "OpenEI: An open framework for edge intelligence," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019.
[71] Q. Zhang *et al.*, "OpenVDAP: An open vehicular data analytics platform for CAVs," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1310–1320.
[72] L. Liu, X. Zhang, Q. Zhang, A. Weinert, Y. Wang, and W. Shi, "AutoVAPS: An IoT-enabled public safety service on vehicles," in *Proc. 4th Workshop Int. Sci. Smart City Oper. Platforms Eng. (SCOPE)*, New York, NY, USA, 2019, pp. 41–47.
[73] H. Stübing *et al.*, "SIM TD: A car-to-x system architecture for field operational tests [topics in automotive networking]," *IEEE Commun. Mag.*,

vol. 48, no. 5, pp. 148–154, May 2010.
[74] (2010). *Deufrako*. [Online]. Available: http://deufrako.org/web/index.php
[75] M. Chevreuil, "IVHW: An inter-vehicle hazard warning system concept within the DEUFRAKO program," in *Proc. e-Saf. Congr. Exhib.*, Lyon, France, 2002.
[76] W. Franz, H. Hartenstein, and M. Mauve, "Inter-vehicle-communications based on ad hoc networking principles. The FleetNet project," Universitätsverlag Karlsruhe Karlsruhe, Karlsruhe, Germany, Tech. Rep., 2005.
[77] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer, "Position-aware ad hoc wireless networks for inter-vehicle communications: The Fleetnet project," in *Proc. 2nd ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2001, pp. 259–262.
[78] A. Festag, H. Fußler, H. Hartenstein, A. Sarma, and R. Schmitz, "FleetNet: Bringing car-to-car communication into the real world," *Computer*, vol. 4, no. L15, p. 16, 2004.
[79] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz, "CarTALK 2000: Safe and comfortable driving based upon inter-vehicle-communication," in *Proc. IEEE Intell. Vehicle Symp.*, vol. 2, Jun. 2002, pp. 545–550.
[80] P. L. J. Morsink *et al.*, "CarTALK 2000: Development of a co-operative ADAS based on vehicle-to-vehicle communication," in *Proc. 10th World Congr. Exhib. Intell. Transp. Syst. Services*, Madrid, Spain, Nov. 2003.
[81] Y. Chen, G. Gehlen, G. Jodlauk, C. Sommer, and C. Görg, "A flexible application layer protocol for automotive communications in cellular networks," in *Proc. 15th World Congr. Intell. Transp. Syst. (ITS)*, New York City, NY, USA, 2008, pp. 1–9.
[82] M. Schulze, G. Nocker, and K. Bohm, "PReVENT: A European program to improve active safety," in *Proc. 5th Int. Conf. Intell. Transp. Syst. Telecommun.*, Paris, France, 2005.
[83] A. Hiller, A. Hinsberger, M. Strassberger, and D. Verburg, "Results from the WILLWARN project," in *Proc. 6th Eur. Congr. Exhib. Intell. Transp. Syst. Services*, 2007, pp. 1–8.
[84] A. Festag *et al.*, "'NoW–network on wheels': Project objectives, technology and achievements," Tech. Rep., 2008.
[85] G. Toulminet, J. Boussuge, and C. Laurgeau, "Comparative synthesis of the 3 main European projects dealing with Cooperative Systems (CVIS, SAFESPOT and COOPERS) and description of COOPERS Demonstration Site 4," in *Proc. 11th IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2008, pp. 809–814.
[86] F. Bonnefoi, F. Bellotti, T. Scendzielorz, and F. Visintainer, "SAFESPOT applications for infrasructurebased co-operative road safety," in *Proc. 14th World Congr. Exhib. Intell. Transp. Syst. Services*, 2007, pp. 1–8.
[87] A. Papathanassiou and A. Khoryaev, "Cellular V2X as the essential enabler of superior global connected transportation services," *IEEE 5G Tech Focus*, vol. 1, no. 2, pp. 1–2, Jun. 2017.
[88] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, "Enhancements of V2X communication in support of cooperative autonomous driving," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 64–70, Dec. 2015.
[89] A. De La Fortelle *et al.*, "Network of automated vehicles: The AutoNet 2030 vision," in *Proc. ITS World Congr.*, 2014, pp. 1–10.
[90] A. Marjovi, M. Vasic, J. Lemaitre, and A. Martinoli, "Distributed graph-based convoy control for networked intelligent vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 138–143.
[91] U. Khan, P. Basaras, L. Schmidt-Thieme, A. Nanopoulos, and D. Katsaros, "Analyzing cooperative lane change models for connected vehicles," in *Proc. Int. Conf. Connected Vehicles Expo (ICCVE)*, Nov. 2014, pp. 565–570.
[92] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models," *Transp. Res. Rec.*, vol. 1999, pp. 86–94, Jan. 2007.
[93] (2018). *Miovision Unveils the World's Smartest Intersection in Detroit*. [Online]. Available: https://miovision.com/press/miovision-unveils-the-worlds-smartest-intersection-in-detroit/
[94] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou, "Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2377–2396, 4th Quart., 2015.
[95] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.
[96] K. Zheng, Q. Zheng, H. Yang, L. Zhao, L. Hou, and P. Chatzimisios, "Reliable and efficient autonomous driving: The need for heterogeneous vehicular networks," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 72–79, Dec. 2015.
[97] J. N. G. Isento, J. J. P. C. Rodrigues, J. A. F. F. Dias, M. C. G. Paula, and A. Vinel, "Vehicular delay-tolerant networks? A novel solution for vehicular communications," *IEEE Intell. Transp. Syst. Mag.*, vol. 5, no. 4, pp. 10–19, Oct. 2013.
[98] J. A. Dias, J. J. Rodrigues, N. Kumar, and K. Saleem, "Cooperation strategies for vehicular delay-tolerant networks," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 88–94, Dec. 2015.
[99] A. Ghosh, V. V. Paranthaman, G. Mapp, O. Gemikonakli, and J. Loo, "Enabling seamless V2I communications: Toward developing cooperative automotive applications in VANET systems," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 80–86, Dec. 2015.
[100] (2018). *Automotive Edge Computing Consortium*. [Online]. Available: https://aecc.org/
[101] J. G. Andrews *et al.*, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
[102] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 546–556, Apr. 2015.
[103] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, 2015.
[104] K. N. Truong, S. N. Patel, J. W. Summet, and G. D. Abowd, "Preventing camera recording by designing a capture-resistant environment," in *Proc. Int. Conf. Ubiquitous Comput.* Springer, 2005, pp. 73–86.
[105] R. T. Ioannides, T. Pany, and G. Gibbons, "Known vulnerabilities of global navigation satellite systems, status, and potential mitigation techniques," *Proc. IEEE*, vol. 104, no. 6, pp. 1174–1194, Jun. 2016.
[106] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, vol. 24, 2016.
[107] S.-Y. Jeong *et al.*, "A study on ROS vulnerabilities and countermeasure," in *Proc. Companion ACM/IEEE Int. Conf. Hum.-Robot Interact.*, 2017, pp. 147–148.
[108] F. J. R. Lera, J. Balsa, F. Casado, C. Fernández, F. M. Rico, and V. Matellán, "Cybersecurity in autonomous systems: Evaluating the performance of hardening ROS," Málaga, Spain, Tech. Rep., vol. 47, 2016.
[109] J. McClean, C. Stull, C. Farrar, and D. Mascareñas, "A preliminary cyber-physical security assessment of the robot operating system (ROS)," *Proc. SPIE*, vol. 8741, May 2013, Art. no. 874110.
[110] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of Networked and Embedded Control Systems* Springer, 2005, pp. 741–765.
[111] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2010, pp. 447–462.
[112] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *Proc. Int. Conf. Internet Things (IOT)*, Oct. 2014, pp. 13–18.
[113] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Secur. Symp.*, San Francisco, CA, SA, 2011, pp. 77–92.
[114] J. Hu *et al.*, "Link level performance comparison between LTE V2X and DSRC," *J. Commun. Inf. Netw.*, vol. 2, no. 2, pp. 101–112, 2017.
[115] M. Raya and J.-P. Hubaux, "Securing vehicular ad hoc networks," *J. Comput. Secur.*, vol. 15, no. 1, pp. 39–68, 2007.
[116] R. G. Engoulou, M. Bellaíche, S. Pierre, and A. Quintero, "VANET security surveys," *Comput. Commun.*, vol. 44, pp. 1–13, May 2014.
[117] Y. Yang, Z. Wei, Y. Zhang, H. Lu, K.-K. R. Choo, and H. Cai, "V2X security: A case study of anonymous authentication," *Pervas. Mobile Comput.*, vol. 41, pp. 259–269, Oct. 2017.
[118] A. M. Malla and R. K. Sahu, "Security attacks with an effective solution for dos attacks in VANET," *Int. J. Comput. Appl.*, vol. 66, no. 22, pp. 1–5, 2013.
[119] B. Yu, C.-Z. Xu, and B. Xiao, "Detecting sybil attacks in VANETs," *J. Parallel Distrib. Comput.*, vol. 73, no. 6, pp. 746–756, 2013.
[120] J. Petit, M. Feiri, and F. Kargl, "Spoofed data detection in VANETs using dynamic thresholds," in *Proc. VNC*, 2011, pp. 25–32.
[121] J. Liu and J. Liu, "Intelligent and connected vehicles: Current situation, future directions, and challenges," *IEEE Commun. Standards Mag.*, vol. 2, no. 3, pp. 59–65, Sep. 2018.
[122] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards IoT-DDoS prevention using edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge 18)*, 2018.
[123] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Springer, 2015, pp. 685–695.
[124] Z. Ning, J. Liao, F. Zhang, and W. Shi, "Preliminary study of trusted execution environments on heterogeneous edge platforms," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 421–426.
[125] D. Li, Z. Zhang, W. Liao, and Z. Xu, "KLRA: A kernel level resource auditing tool for IoT operating system security," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 427–432.
[126] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. Reddi, "MAVBench: Micro aerial vehicle benchmarking," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2018, pp. 894–907.
[127] Y. Wang, L. Liu, X. Zhang, and W. Shi, "HydraOne: An indoor experimental research and education platform for CAVs," in *Proc. 2nd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*. Renton, WA, USA: USENIX Association, 2019.

ABOUT THE AUTHORS

Shaoshan Liu (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of California at Irvine, Irvine, CA, USA, in 2010.

He was a Founding Member of Baidu USA, Sunnyvale, CA, USA, and the Baidu Autonomous Driving Unit, where he was in charge of system integration of autonomous driving systems. He is currently the Founder and CEO of PerceptIn, Fremont, CA, USA, a company focusing on providing visual perception solutions for autonomous robots and vehicles. He has authored or coauthored more than 40 high-quality research papers. He holds more than 150 U.S. international patents on robotics and autonomous driving. He is also the Lead Author of the best-selling textbook *Creating Autonomous Vehicle Systems*, which is the first technical overview of autonomous vehicles written for a general computing and engineering audience. His current research interests include computer architecture, deep learning infrastructure, robotics, and autonomous driving.

Dr. Liu co-founded the IEEE Special Technical Community on Autonomous Driving Technologies to bridge communications between global autonomous driving researchers and practitioners and serves as the Founding Vice President. He is an ACM Distinguished Speak and an IEEE Computer Society Distinguished Speaker.

Liangkai Liu received the B.S. degree in telecommunication engineering from Xidian University, Xi'an, China, in 2017. He is currently working toward the Ph.D. degree at Wayne State University, Detroit, MI, USA.

His current research interests include edge computing, distributed systems, and autonomous driving.

Jie Tang (Member, IEEE) received the B.E. degree in computer science from the National University of Defense Technology, Changsha, China, in 2006, and the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2012.

She was a Visiting Researcher with the Embedded Systems Center, University of California at Irvine, Irvine, CA, USA, and a Research Scientist with Intel China Runtime Technology Lab, Beijing. She is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Her current research interests include research on computer architecture, autonomous driving, cloud, and run-time system.

Dr. Tang is a Founding Member and the Secretary of the IEEE Computer Society Special Technical Community on Autonomous Driving Technologies.

Bo Yu (Member, IEEE) received the B.S. degree in electronic technology and science from Tianjin University, Tianjin, China, in 2006, and the Ph.D. degree from the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2012.

He is currently the CTO of PerceptIn, Fremont, CA, USA, a company focusing on providing visual perception solutions for robotics and autonomous driving. His current research interests include algorithm and systems for robotics and autonomous vehicles.

Dr. Yu is a Founding Member of the IEEE Special Technical Community on Autonomous Driving.

Yifan Wang received the B.S. degree in electronic information engineering from Tianjin University, Tianjin, China, in 2014. He is currently working toward the Ph.D. degree at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

His current research interests include computer architecture, edge computing, and connected and autonomous vehicles.

Weisong Shi (Fellow, IEEE) received the B.S. degree in computer engineering from Xidian University, Xi'an, China, in 1995, and the Ph.D. degree in computer engineering from the Chinese Academy of Sciences, Beijing, China, in 2000.

He is currently the Charles H. Gershenson Distinguished Faculty Fellow and a Professor of computer science with Wayne State University, Detroit, MI, USA. His current research interests include edge computing, computer systems, energy efficiency, and wireless health.

Dr. Shi was a recipient of the National Outstanding Ph.D. Dissertation Award of China and the NSF CAREER Award.