Collaborative Autonomous Driving: Vision and Challenges

Zheng Dong[†], Weisong Shi[†], Guangmo Tong[‡] and Kecheng Yang[§]
[†]Department of Computer Science, Wayne State University
[‡]Department of Computer and Information Sciences, University of Delaware
[§]Department of Computer Science, Texas State University

***Abstract*—This paper discusses challenges in computer systems research posed by the emerging autonomous driving systems. We first identify four research areas related to autonomous driving systems: real-time and embedded systems, machine learning, edge computing, and cloud computing. Next, we sketch two fatal accidents caused by active autonomous driving, and uses them to indicate key missing capabilities from today's systems. In light of these research areas and shortcomings, we describe a vision of digital driving circumstances for autonomous vehicles and refer to autonomous vehicles as "clients" of this digital driving circumstance. Then we propose a new research thrust: collaborative autonomous driving. Intuitively, requesting useful information from a digital driving circumstance to enable collaborative autonomous driving is quite sophisticated (e.g., collaborations may come from different types of unstable edge devices), but it also provide us various research challenges and opportunities. The paper closes with a discussion of the research necessary to develop these capabilities.**

***Index Terms*—autonomous driving, real-time system, machine learning, edge computing, cloud computing**

I. Introduction

In 1885, Karl Benz developed the first modern automobile, which was powered by a single cylinder four-stroke engine with 2/3 horse power [1]. More than 100 years later, the automobile has now become much improved - the engine of a Tesla Roadster 2.0 can provide 1,242 horse power and can accelerate from 0 to 60 mph in 1.9 seconds, with a top speed of 250 mph [2]. However, human drivers cannot drive cars at such top speeds in their daily life due to safety issues. Speed limits (e.g., along one stretch of highway in Pennsylvania a human driver can drive at most 65 mph) guarantee human drivers enough time to react to changes in the roadway or flow of traffic, making it easier for human drivers to stop their cars if needed. In other words, as cars are the major method for land transportation in the US, their travelling speed is constrained by the reaction time required by human drivers. Do we need faster travelling speeds on highways? The answer is Yes. Researchers are making every effort to raise the speed of the car: inventing new powerful engines and building expressways across the country. The question is, will the current speed limits continue to impact our daily life in the next 100 years?

With the development of multi-core technologies, automobile manufacturers are starting to discover the benefits of doing more computing and analytics on the devices themselves using powerful embedded computing platforms. This on-device approach provides new possibilities of designing autonomous driving systems to take the place of human drivers by deploying and verifying multiprocessor implementations of real-time control systems in the car. Examples from industry include Model 3, X and S vehicles at Tesla [2]; Apollo at Baidu [3]; Waymo - formerly the Google autonomous car project which stands for a new way forward in mobility [4] now, and some other autonomous vehicles developed by Ford, Toyota, Mercedes, etc. In academia, examples include project ALV at CMU [5], EUREKA Prometheus Project at Bundeswehr University Munich, Mcity at University of Michigan [6] and HydraOne [7] at Wayne State University. All of these projects study a different mix of issues in autonomous driving, and a different blend of short-term and long-term goals. However, in general, they represent a bright future to deliver the control of cars from human drivers to the intelligent embedded systems and make autonomous driving a reality.

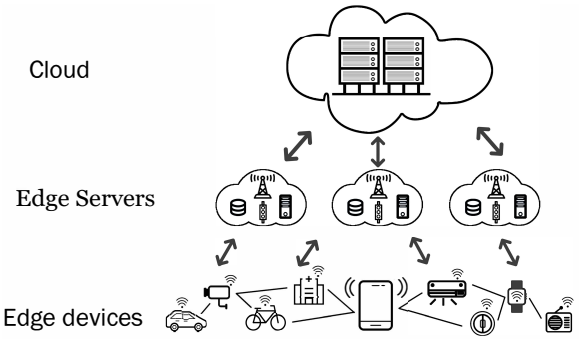

Fig. 1. Digital driving circumstance for autonomous vehicles. There are two types of edge devices: mobile edge device (e.g., mobile phones, bikes and cars) and fixed edge device (e.g.,cameras and loudspeakers). The peer-to-peer communication is enabled among the edge devices, but the quality is not stable due to the limited communication range. The communication between edge devices and edge servers is much better since the edge servers consist of better computing platforms and cellular base stations, which provide a stable cellular network. However, each edge server only captures and processes local information around its location. In order to get a global view of the entire driving circumstance, the cloud collects the processing results from all the edge servers. Thus, the computer system research related to autonomous driving systems includes both edge computing and cloud computing. In other words, autonomous driving will be the crucible to which many different research areas are fused.

Benefits of autonomous driving are obvious. Increased travelling speed and more free time with little attention on the driving circumstance are at the top of the list. We can turn roughly 204 hours we spend each year commuting into time spent surfing the internet, studying or even playing games. Autonomous driving would also bring with it the benefits of decreased traffic and reduced emissions, but the safety issue is still the major concern impeding the wide adoption

of autonomous driving cars. The goal of this work is to introduce the challenges in computer systems research posed by autonomous driving and discuss how to construct a safe autonomous driving circumstance in the future. We begin by introducing real-time operating systems, which are used to operate the car correctly through efficient real-time task scheduling, and examining its relationship to the closely-related fields of edge computing and cloud computing. Next, we sketch a typical accident scenario and ask why the existing techniques cannot avoid an accident. In light of these research areas and shortcomings, we describe a vision of digital driving circumstances for autonomous vehicles, which accompanies it everywhere and mediates all interactions with the edge computing elements in its surroundings. The car is likely to be implemented by a powerful computer with four wheels. We refer to an autonomous car as the "client" of its digital driving circumstance. From that starting point, we delve deeper into some key research problems.

II. Related Fields

The computing industry recently experienced a major shift in the computing paradigm from the cloud to the edge. This shift has necessitated the adoption of new programming models, algorithms, and analysis methods to fully exploit the computing capacity of multi-core chips deployed on the edge. Fig. 1 shows the digital driving circumstance for autonomous vehicles and in this section, we will introduce the computer system research areas related to autonomous driving systems one-by-one in detail.

A. Heterogeneous Computing Platform

In order to achieve a greater computing capability to support increasingly computation-intensive autonomous features while maintaining an affordable cost in terms of energy and dollars, heterogeneous computing hardware should be deployed in autonomous vehicles. In a company with traditional CPU-like computing elements for general computations, special-purpose computing elements, such as graphic processing units (GPUs), digital signal processors (DSPs), and customized processors by FPGAs, are able to provide powerful computing capabilities for certain kinds of workloads in a cost-efficient way. Furthermore, even for the same computing purpose (i.e., targeting the same kind of workload), a combination of low-performance, low-cost computing elements and high-performance, high-cost ones may enable an even higher cost efficiency.

As we will see next, machine learning and computer vision algorithms are heavily deployed to enable the autonomous feature and therefore create a great portion of workload to be supported by the computing system for autonomous vehicles. GPUs are very powerful in processing such workloads, but managing them in a predictable manner can be challenging. Fortunately, significant research effort has been conducted in the direction of predictable GPU usage, e.g., [8]–[21], which will enable the adoption of GPUs in time-critical and safety-critical tasks in an autonomous driving system.

To further integrate heterogeneous computing elements in addition to GPUs, a recently ratified standard, called OpenVX [22], can be adopted, and efforts for applying OpenVX in time-critical systems have been made in the last few years [23]–[28]. In OpenVX, certain computations are encapsulated in a *node*, which has designated input and output parameters bound to shared (between nodes) data objects. Dependencies, or *edges*, are derived accordingly. As a result, freedom of executing individual nodes on different types of computing elements are provided, enabling the computations in all nodes to be performed on their respective most-effective hardware.

B. Machine Learning related Real-time Tasks

Machine learning is an essential part of today's autonomous vehicle system in which the learning tasks can be generally classified into two categories: tasks for driving decision making and tasks for specific scientific purposes.

1) Tasks for Driving Decision Making: A successful driving system is a fundamental requirement in all kinds of autonomous vehicle systems, and it critically relies on the correct driving decisions. The inputs of the driving system are obtained from sensors, such as odometry, range sensors, various cameras and laser scanners, and they are then processed for identifying the surrounding environment to make driving decisions. Such identifications are mainly made by object detection and tracking - the former tells what the objects are; the latter tells where they would be.

Object detection on an input image typically consists of two steps [29]: region of interest (ROI) extraction and object classification. ROI can be extracted either by the sliding window approach, which shifts a detector over the image, or the selective search approach, which extracts target locations through segmentation [30]. Object classification has drawn tremendous attention in the computer vision community, and the state-of-the-art solutions are deep learning approaches design based on convolutions neural networks. Among the object detection tasks, person detection is particularly important but challenging as pedestrians can have diverse poses and appearances [31]. Semantic segmentation aims to assign each pixel a class label, and it can be viewed as fine-grained object detection. Traditionally, semantic segmentation is formulated as an inference problem of conditional random fields, and the co-occurrence of objects in different classes can be either independent [32] or dependent [33]. With the advance of deep learning especially convolutional neural networks, various architectures have been proposed, where the CITYSPACE [34] dataset is a popular benchmark for such problems.

Tracking aims to detect the state of a certain object over a time period, and it is an important task as being aware of the future trajectory of other traffic participants is necessary for avoiding potential collisions. In one branch, the tracking problem can be viewed as predicting the next state of all the points in a random field. Early methods solve this problem by data association in which the maximum-a-posteriori solution is obtained by solving a min-cost flow problem [35], while

the state-of-the-art solutions utilize the continuous energy minimization approaches [36], [37]. In another popular branch, tracking-by-detection solves this problem by first detecting the object and tacking the targets through a sequence of detection results. One drawback of such methods is that they heavily rely on the accuracy of detection, and their results can rarely be generalized to unseen objects. One promising tracking method is to combine multiple factors to maximally utilize the obtainable information, and such factors can be texture, shape, object detection and localization, stereo depth, and trajectory. From a higher level, one could jointly predict trajectories and other interested quantities such as pose, positions, stereo depth and poses [38]. Pedestrian tracking is again important but difficult as the movement of people is less predictable. Andriluka *et al.* [39] proposed a method that combines the detection and tracking, and jointly models them with a hierarchical Gaussian process latent variable model, and they later extend the idea using the hidden Markov model for tracking over long time periods.

2) Tasks for Specific Scientific Purposes: While using autonomous cars for passenger-carrying has not been popularized due to safety concerns, autonomous vehicles are now widely used in different domains of science and engineering in which their main task is to collect and more importantly analyze data from surrounding environments. For example, autonomous underwater vehicles (AUV) are able to work in an extreme environment such as deep hydrothermal vents to polar ice sheets, and they have been widely used in marine geoscience [40]. In agriculture applications, unmanned aerial vehicles (UAVs) are used as a sensing technology for facilitating precision agriculture [41], and they can also be used for tracking the path of plant pathogens [42]. Machine learning in such tasks is highly domain-dependent and often customized for a specific purpose. Furthermore, the driving system in such scenarios is different from highways or urban environments, and they have to consider various factors that are not commonly seen in the benchmark dataset. For example, autonomous underwater vehicles may require a successful detection of aquatic life or reefs but not pedestrians.

C. Real-Time Operating System

In autonomous driving, control decisions must be made and committed in milliseconds to enable responsive and desirable behaviors of the vehicles. Therefore, the operating system (OS) that allocates computing resources to the tasks that implement the autonomous functionalities must be a *real-time* one. That is, the temporal metrics, behaviors, and properties of the system need to be predictable, analyzable, and certifiable. With the ever increasing integration of functionalities on the shared hardware platform, a few key features are expected in a real-time operating system (RTOS) for autonomous driving.

1) Mixed Criticality: Because of the size, weight, and power (SWaP) constraints and need for cost efficiency, the number of dedicated and isolated hardware computing elements cannot match that of the explosively expanding functionalities. As a result, it is inevitable for computing resources to be shared by multiple functionalities, which are potentially of different *criticalities*. Thus, managing mixed-criticality tasks in RTOS is expected for autonomous driving. Tasks of different criticalities are provisioned under different assumptions, reflecting different degrees of assurance. In particular, criticality does not necessarily imply urgency and consequently does not necessarily dictate priority as well.

The design of a mixed-criticality real-time system was first proposed by Vestal [43] by introducing multiple WCET estimates of multiple levels of assurance. Subsequently, a large body of research efforts addressing mixed-criticality real-time systems has been conducted. [44] is a comprehensive review of this topic specifically and has been updated annually. As of now, 514 references have been cited in [44].

2) Real-time Offloading: Leveraging remote edge resources to facilitate the development of new simultaneous localization and mapping [46] (SLAM) algorithms in an autonomous driving system is of great importance. SLAM algorithms are used to correct the system's error accumulation or drift in a real-time fashion. To ensure functional correctness of the operations, it is critical to ensure the bounded response times of SLAM-induced computations along with other workloads in the system. Handling this situation in real-time requires a great amount of time, computation power, and energy. Using a battery-powered vehicle alone to process these computations is infeasible due to its limited computational power and stringent energy constraints.

To accelerate the computation and save the energy of the embedded computing platform, our approach is to establish an energy efficient offloading framework for selectively offloading computations to remote resources (e.g., edge servers) while ensuring timing predictability (i.e., ensuring applications to have provably bounded response times). Note that although we focus on the SLAM algorithm as an example, our designed framework is applicable to all relevant algorithms (e.g., object recognition) in an autonomous vehicle where achieving both timing predictability and energy efficiency is necessary [47].

3) Synchronization Protocols: In addition to the computing elements that are being scheduled to support task computations, certain computations might also require other shared resources, which can be hardware (e.g., I/O devices) or software (e.g., shared data objects). Similarly, the computing elements are managed by the scheduler in the RTOS, and these additional shared resources are often managed by synchronization protocols. In addition to maintaining proper consistence and desired access orders as in a general-purpose OS, synchronization protocols in RTOS have to provide provably bounded waiting time for every resource access request. In particular, bounding the duration of *priority-inversion blocking (pi-blocking)*, where a lower-priority task holding a shared resource prevents a higher-priority task from executing on available processors, is crucial.

To this end, the classic priority inheritance protocol (PIP) [48], priority ceiling protocol (PCP) [49], and stack-based resource allocation protocol (SRP) [50] have been proposed and widely well-received for uniprocessor scenarios. There

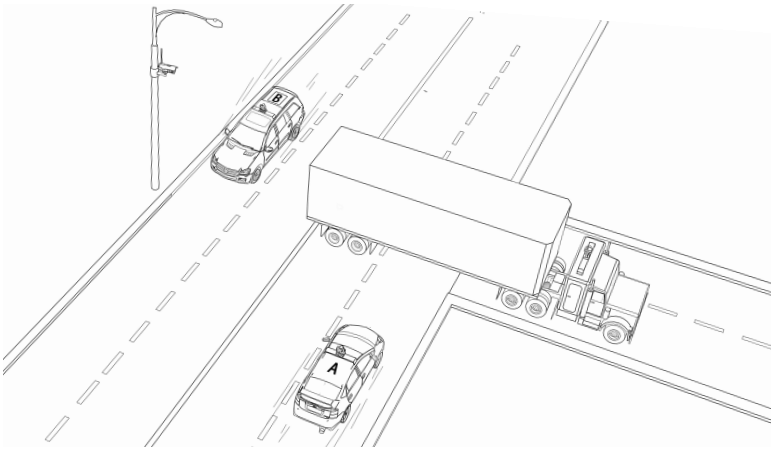

Fig. 2. A Tesla Model S (denoted by letter A) with the Autopilot system activated was involved in a fatal crash in 2016, the first known fatality in a Tesla where Autopilot was active. The vehicle was on a divided highway with Autopilot engaged when a tractor trailer drove across the highway perpendicular to the Model S. Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of the impact caused the Model S to pass under the trailer, with the bottom of the trailer impacting the windshield of the Model S. In a tweet, Tesla CEO Elon Musk said that the vehicle's radar did not help in this case because it "tunes out what looks like an overhead road sign to avoid false braking events" [45].

are also their variants to cope with multiprocessor scheduling, e.g., [51]–[53]. More recently, the flexible multiprocessor locking protocol (FMLP) [54] was designed to treat short and long resource requests differently; the $\mathcal{O}(m)$ locking protocol (OMLP) [55] and OMLP family [56] are able to achieve asymptotically optimal pi-blocking results; the real-time nested locking protocol (RNLP) family [57]–[59] specifically addresses fine-grained nested resource requests.

D. Edge Computing

In the connected and collaborative autonomous driving ecosystems we are aiming to build, computing capacity at local *edge servers* [60], such as processing elements reside in drive way infrastructures, must be leveraged, as shown in Fig. 1. Because autonomous vehicles are expected to moving between edges servers that are fixtures, dynamic variations of the computation workload at edge from each edge server's point of view are also expected and should be highlighted.

As the geographic location of an autonomous vehicle is changing, the connection between it and a certain edge server might be weakened or even lost. Then, it is clearly desirable for the vehicle to be re-connected to a "closer" edge server to offload its computations to the new edge server. On the other hand, from an edge server's perspective, it is equivalent to having certain tasks (of vehicles driving away) leaving and other tasks (of vehicles driving in) joining this particular local computation system at edge dynamically.

Such a dynamic workload may jeopardize most classic analysis and certification techniques for system temporal correctness. Fortunately, approaches specifically addressing such a dynamic workload also exist in the literature. The well-received proportional share [61], [62] provides a seminal solution for the allocation of a dynamic workload. Later research efforts further investigated adaptive approaches to address dynamic workload under various settings, e.g., [63]–[67].

E. Cloud Computing

While keeping the computations at edge may have the advantage of better responsiveness and predictability, it is also limited by the maximum computing capability of each edge server and potential lack of a global knowledge for collaborations between edge devices. Therefore, as shown in Fig. 1, certain computations might have to be done in the cloud, and then the results would be transmitted to the edge.

Although such computations often benefit from the powerful computing capability in the cloud and obtain superior *average* performance, it can be challenging to provide worst-case guarantees for each individual end result at edge. Edge servers and devices typically have no authority to control the resource allocation in the cloud, so any desired guarantees have to be propagated in a top-down fashion. In particular, in order to guarantee the results available at edge in a timely manner, frameworks to support a compositional real-time systems approach should be applied and implemented from the cloud computing service providers all the way down to each edge device in a hierarchical manner. Existing work on this topic, e.g., [68]–[80], may be adopted and revised to enable cloud computing even in certain time-critical components for autonomous driving.

III. Example Scenarios

What is the most critical technical issue restraining the development of autonomous vehicles? To answer this question, we describe two real scenarios below. Note that the similar accidents have appeared twice in the past few years.

A. Two fatal accidents

Fig. 2 shows a fatal crash in 2016, where a Tesla Model S did not notice a big rig or the trailer "against a brightly lit sky" and passed under the trailer. The detailed description of this accident is explained in the figure's caption. Unfortunately, a similar accident happened again in 2019. A red Tesla Model

3 was driving south in the right lane of State Highway 441 in Delray Beach and a truck pulled out of a private driveway on the right side of the road, heading across the highway and intending to turn left, going north. The truck slowed as it crossed the southbound lanes, blocking the Tesla's path. The car struck the trailer at 68 mph without making any evasive maneuvers. It passed under the trailer, ripping off its roof and killing the driver. In both cases, a Tesla running on Autopilot on a Florida highway struck a truck cutting across its path, killing the Tesla's driver [81].

B. Missing capacities

These fatal accidents embody many key issues in autonomous driving systems.

First, we can see that it is not sufficient to make driving decisions reliably based on the results of images processing, which are captured by an "individual" vehicle from a single direction. The Tesla Model S sedan hit the bottom of a semitrailer as it passed underneath since the car did not recognize the semitrailer by itself. Thus, even if an autonomous driving car can identify the objects in its way in most cases, it cannot guarantee 100% reliability. Fundamentally, it is challenging to ensure real-time reliability in current autonomous driving systems, preventing such systems from being legally certifiable and thus causing safety to be a major concern.

Intuitively, if vehicle-to-vehicle communication is enabled between the car and the truck, the truck can share its real-time information (i.e., specifications, location and speed) with the car, and in this case, the car can recognize the truck quickly. However, vehicle-to-vehicle communication is not stable, esoterically in high-speed driving circumstances. Considering the limited communication range, vehicle-to-vehicle communication is valuable for autonomous driving systems if its communication delay and reliability are significantly improved in the future.

Another intuitive idea is to resolve the safety issue through "collaborative autonomous driving". Accidents occur because current autonomous vehicles perform autonomous driving by relying on their individual observations, i.e., driving decisions are made based on the results of image processing, and images are captured by themselves from a single direction. Thus, if the information contained in the images is not enough to fully describe the driving circumstance, the driving decisions may be incorrect. We believe the most fruitful approach to comprehensively describe the driving circumstance is a collaborative approach, i.e., by collaboratively utilizing multiple cameras on the edge around the autonomous vehicle, the driving circumstance can be observed from multiple directions and everything in the driving circumstance can be recognized correctly. In light of different scenarios, the car can collaborate with two types of edge cameras: mobile edge cameras and fixed edge cameras. All the autonomous driving vehicles are equipped with mobile cameras that travel around and recognize the objects appearing in their driving circumstance. For example, in Fig. 2, car A and car B are travelling in opposite directions. Both of them should recognize the truck, but car A fails to do so. If car B can recognize the truck in advance and share this information with car A, car A may be able to stop in time. Similarly, the fixed edge camera can also perform object recognition algorithms to identify the truck in Fig. 2 and remind car A to be aware of the truck. In both scenarios, the recognition algorithms are performed on the car and edge cameras. How to correctly define the execution behavior of these real-time workloads and efficiently schedule all the tasks in real-time are fundamental issues to realize "collaborative autonomous driving". Therefore, based on the discussion in this section, we will talk about the research challenges and opportunities next.

IV. Research challenges and opportunities

Practical realization of autonomous driving system will require us to solve many difficult design and implementation problems. In light of the discussion in earlier sections, we now look at some of these problems at the next level of detail. Our goal is only to convey an impressionistic picture of the road ahead and this specific set of topics is merely a sampling of the problem space, presented in no particular order.

In this section, we assume that each autonomous car is immersed in a digital driving circumstance (as shown in Fig. 1) that accompanies it everywhere and mediates all interactions with the edge computing elements in its surroundings. The car is likely to be implemented by a powerful computer with four wheels. We refer to an autonomous car as the "client" of its digital driving circumstance, even though many of its interactions may be yielded with other clients (autonomous cars) in the same area. Intuitively, the collaboration from the digital driving circumstance to enable collaborative autonomous driving is quite sophisticated (e.g., collaborations may come from different types of edge devices), but it also provides us various research opportunities.

A. Virtual Driving Circumstance Construction

For an autonomous driving system, huge volumes of data generated by edge devices may congest at the autonomous vehicle at some specific locations. It will be almost impossible to determine which information will help rather than hinder the autonomous vehicle in real-time. For example, suppose the vehicle arrives at an intersection. Should the system inform the vehicle that (*i*) the traffic light has turned green, (*ii*) a car is coming in the opposite direction, which may need to be yielded, or (*iii*) a pedestrian is in the crosswalk? The correct choice will depend on the destination of this vehicle.

For proactivity to be effective, we envision a virtual driving circumstance to be constructed in advance that can collect, maintain and deliver the driving circumstance information to the autonomous vehicles.

1) Vehicle to Edge Communication: As we discussed in Sec. II, numerous edge devices and edge servers coexist around the autonomous vehicle, generating and processing information from the driving circumstance. When the vehicle detects an edge device or an edge server, it first requests information from it. Communication with an edge device is

via short-range wireless peer-to-peer technology, and communication with an edge server is via long-range cellular network. Consider the example given in Fig. 2. If car A detects either car B or the fixed camera, car A can request information about the driving circumstance detected from their perspectives. In this case, even if car A does not detect the truck, the truck will be put into car A's virtual driving circumstance by car B or the fixed camera and the crash can be avoid. Edge servers can serve as the vehicle's networking gateway to the Internet then the virtual driving circumstance can cover a larger range for this autonomous vehicle. When the vehicle leaves the edge device, it can cache the data in its disk and share the information with other vehicles in the future or discard the data to empty its storage. Vehicle-to-Edge Communication opens up many important research questions. Here are some examples:

- How to discover the surrounding edge devices? There are many proposed service discovery mechanisms such as JINI, UPnP, and BlueTooth proximity detection, so which one is best suited for our purpose [82]?
- How to balance workloads among edge devices? Is workload allocation based on an admission control approach or a best-effort approach?
- What is the criteria to save or discard the information captured from the driving circumstance?
- How many fixed edge devices are needed to construct a stable virtual driving circumstance in a specific area?
- In typical situations, how much advance information does an edge device need to issue as a virtual driving circumstance constructor with minimum delay? Is this on the order of seconds, minutes or tens of minutes?

2) Real-time Object Tracking: Through vehicle-to-edge communication, edge devices and edge servers are linked into the vehicle's virtual driving circumstance. However, objects, such as traditional vehicles and pedestrians, may not be equipped with communication devices, which cannot be linked directly. In order to construct a practical and complete virtual driving circumstance, it is necessary to track them using cameras deployed on the edge. There are two types of cameras that can be used to perform object tracking [83].

First is the road video surveillance system. Cameras are deployed at intersections in the road network, which are equipped with video surveillance system. Every object passing an intersection with a video surveillance system can be detected, and the corresponding video is captured. A road video surveillance system can provide stable tracking services, but the number of cameras is not large and the coverage of the monitored area is not very wide.

Second is the autonomous driving fleet. Autonomous vehicles are equipped with cameras that can be used to track objects in the way. Every object appearing in the driving circumstance can be detected and tracked. The number of autonomous vehicles is large, but the appearance of each autonomous vehicle is not predictable.

Both types of cameras are important for object tracking to construct the virtual driving circumstance. The mobile cameras can be used to tracking objects in the hidden areas where the surveillance system cannot reach. This complementary relationship between the two types of cameras yields a straight forward research question:

- Considering the dynamics of the autonomous driving fleet, how many fixed cameras are needed in a specific area to build a stable tracking system?

3) High-Definition Map: Traditionally, maps in vehicles are used mainly for navigation purposes. However, the combination of edge devices and on-board HD maps is a very promising approach to help us enable autonomous driving [84]. Vehicles can obtain precise position and road information from the on-board part of the HD map system in real time. Then, a highly dynamic and rapid-changing virtual driving circumstance is developing that enables exact and real-time images of the cars' surroundings. Much infrastructure information can be obtained directly from an HD map, e.g., the traffic lights and stop signs. Some research questions follow:

- One hour of drive time produces up to one terabyte of data. Where shall we store the data?
- For real-time requirements, latency must be lower than 10 ms. How to make an offloading decision to ship the computation from the local device to remote edge devices?
- Data transmission is also a problem. Although today's available LTE (4G) allows data transmission at 100 Mbps, 2.2 Gbps is required.

B. Timing Correctness

Autonomous driving vehicles are safety critical systems. With current technology, very conservative estimates concerning the usage of these computing resources must be made to guarantee the system's temporal correctness at runtime. Such conservation could easily negate the processing power of any additional cores. To enhance the utilization of the multi-core platform while guaranteeing the temporal correctness of the system, computing resources need to be judiciously allocated to workloads in a predictable way. For example, in autonomous driving systems, the actions of detecting a stop sign and then performing the braking action must be completed within a certain time window (e.g., meeting deadlines) otherwise, a traffic collision might occur. The answer to whether "the braking action" can meet its deadline at runtime depends on the real-time scheduling algorithm. Different scheduling algorithms may yield different completion times for tasks scheduled on the multi-core platform. Thus, the development of smart real-time scheduling algorithms along with the corresponding timing validation techniques becomes critical to enhance the computing capability of autonomous driving vehicles with guaranteed temporal correctness [85].

C. Machine Learning in Autonomous Driving

Despite fruitful outcomes from the computer vision community, machine learning tasks still present a few challenges regarding ensuring a successful autonomous driving system. One the one hand, meeting the real-time requirements is

necessary for perfect driving safety, but such requirements are often overlooked by the existing designs. On the other hand, the collaborations between different units, such as vehicles and sensors, are crucial for achieving effective distributed systems, but they have not been widely considered in the current machine learning solutions. Finally, the nature of machine learning makes itself incapable of dealing with unseen scenarios. In what follows, this paper identifies several challenges and opportunities.

1) Flexible Inference Time.: The inference time of object detection or tracking is critical for avoiding potential accidents especially with the sudden emergence of pedestrians and obstacles. Fast inferences have been considered for various convolutions neural networks, and the goal in the existing works is to minimize the inference time without sacrificing the detection accuracy. However, a realistic case might be that the decision must be made by a deadline due to specific emergencies, and therefore, the object detection has to be done in a few seconds or even million-seconds even if at the cost of a decrease in learning performance. Considering the example in Fig. 2, we would prefer a coarse but fast detection over an accurate but slow one. A trade-off between the inference time and learning effect can be achieved through several methods. In one branch, one can hold a collection of separately trained object detection methods from simple to complex, and then select an appropriate one in a certain scenario. Alternatively, one could design one method such as a deep neural network with exits having different inference complexities, and make an early exit if a fast inference is needed. One advantage of doing so is that the exits can be trained concurrently for ensuring overall performance. In another branch, one could design a set of detection algorithms among which some are responsible for coarse classification while others are in charge of fine-grained detection. Such frameworks would call coarse methods in emergent cases in order to meet deadlines. In the example in Fig. 2, in order to reach a break-hitting decision quickly, the Tesla might only need to identify if there was another obstacle in front of it but not need to know if that was a trailer or car.

2) Collaborations for Resource Allocation.: In an autonomous driving system, vehicles, edge devices and the cloud have different levels of computation capacity, and the resource allocation among them is surprisingly not needed. First, a heavy machine learning task initialized by a vehicle may require a powerful computing platform that is only available in nearby edge devices or that an uncommon machine service is only available in certain devices such as the cloud while such a service might be called by individual devices. In such a case, coordinating the execution of such tasks would require a collaborative schedule, which is complicated by the uncertainty caused by the communication failures and the rapid movement of the vehicles and other participants. Second, if one would additionally add real-time requirements, making all the tasks meet their deadlines is a very challenging problem. Finally, the challenges concerning resource allocation are also posed by the priority of the tasks. For an autonomous vehicle running on a highway to detect air pollution, tasks for ensuring safe driving have the highest priority, and they should be first scheduled if the on-vehicle computing resource is limited. For the above issues, effective solutions with performance guarantees, such as provable inference time or a schedulability test, are highly desired but not trivial to design and analyze.

3) Collaborations between Dependent Tasks.: Collaboration is also needed by the machine learning tasks that take input from different devices. Taking the example in Fig. 2 again, a sensor with a camera on the street light would be able to provide an image of the trailer from another angle, and utilizing the information from different devices could facilitate the object detection. An immediate but non-trivial problem is how such collaborations can be achieved. One choice is that we could collect all the required inputs and process such data by a center device. For example, the sensor on the street light can first send the collected image to the coming Tesla, and the Tesla then performs object detection using the received image together with the data collected by itself. Such solutions can utilize all the available data in a centralized manner, but one of their disadvantages is that the data transmission can be an issue especially for the case when the dataset is large and, meanwhile, the required task is emergent. Alternatively, we could let each device process its data locally and somehow obtain a partial result which will be sent to the target devices for global analysis. Under such a framework, the senor on the street light in Fig. 2 would first process the image it receives and send the result to the Tesla, and such a result could be: *from my view, there is a trailer in front of you*. Taking into account all the received partial results as well as its local result, the Tesla will make an enhanced object detection. Such methods can ease the burden in data transmission as only the results are shared between the devices, but they can lack a global view of the surrounding environment because the received partials results are obtained through several local views, and possibly none of them is correct.

4) Generalization to Unseen Scenarios.: For most of the common tasks such as object detection or tracking, the state-of-the-art solutions on benchmarks such as KITTI [86] and Caltech-USA [31] have achieved promising performance in terms of both the accuracy and inference time. However, the feasibility of the existing algorithms is still limited because the *accidents* are not in benchmarks. Indeed, such a problem is ubiquitous as machine learning relies on the fundamental assumption that the training data should be somehow related to the instance of interest. This issue is more severe in machine learning tasks for certain scientific purposes as they are exploring the environment that is not only unknown to the benchmarks but also to humans. One straightforward solution is to construct benchmarks or simulation methods that can comprehensively catch real driving environments to train improved machine learning algorithms, and it aims to strengthen the ability of an individual vehicle. However, such a method would inevitably increase the complexity of the algorithm, resulting in a longer inference time, which is not necessary in most cases as accidents are rare. Another

arguably better solution is to seek help from other participants in the system, such as edge devices and the cloud, when an unseen scenario occurs. That is, the capacity of an individual device remains unchanged, and the improvements are made by collaboration, which enables a seamless connection between the existing machine learning algorithms and the extra efforts to deal with rare cases.

D. Navigation

While vehicle navigation has been a classic and widely-studied problem for decades, the problem of navigation specifically for connected and collaborative autonomous driving can bring new features and hence challenges to be addressed. For example, if the navigation decisions are based on *typical* estimates on the delay for each route, some vehicles might encounter cascaded worst or near-worst scenarios and, therefore, are unable to provide the certain level of predictability that is needed for the ecosystem to be reliably collaborative. On the other hand, using more deterministic and predictable estimates, such as the worst-case delay, may enable a proper and reliable global resource allocation for each edge server; however, the fastest route the navigation system obtained based on these estimates may not be the actual fastest one in the dominant cases, resulting in unacceptably poor average-case navigation performance.

One potential avenue to mitigate this dilemma could be adopting the routing model with *multiple delay estimates* in the problem of *rapid routing with worst-case bound*, proposed by Baruah [87], where both typical and a worst-case estimates on the routing delay are taken into account in the routing problem. Nonetheless, to be ultimately applicable to connected and collaborative autonomous driving, many issues beyond the problem described in [87] must be considered, for example:

- While [87] focused on single-source single-destination routing, simultaneous routing for multiple pairs of source and destination may be desired.
- Connected autonomous vehicles are the to-be-navigated entities that need the delay information, but at the same time, are also part of the traffics that create the delay information. Can this relationship be further explored, resulting in even more superior routing models and approaches?
- What if such estimates vary periodically? This might capture the fact that the estimates on the delay for a certain route may vary at different times in a day e.g., 4 am v.s. 4 pm.

E. Privacy and Security

Privacy and security, already a thorny issue in edge computing and cloud computing, is greatly complicated by autonomous driving. The techniques mentioned above, such as localization, synchronization, real-time tracking, and peer-to-peer collaborations are advancing on a continuous basis. As a "client" becomes more dependent on the digital driving circumstance, it becomes more knowledgeable about that client's locations, behavior patterns and habits. Exploiting this information is critical to provide reliable autonomous driving services. At the same time, unless use of this information is strictly controlled, revealing individual's information to unknown sources may result in identity theft.

In the digital driving circumstance, data collection and processing are pushed to the edge. Edge computing helps protect user privacy by anonymizing, analyzing, and keeping the data at the source rather than sending identifiable information to the cloud. Privacy and trust in autonomous driving systems are likely to be enduring games. We believe there are many research topics in this area, for example:

- What are the authentication protocols best suited to autonomous driving?
- How often should the system empty the out-of-date information?
- What kinds of useful information can be uploaded to the cloud? What kinds of data should be discarded after processing?

F. Energy Management

Most of the autonomous vehicles are powered by batteries. There is growing consensus that advances in battery are achieving a technological plateau - in order to improve the batteries' efficiency, higher levels of the system must also be involved. How does one involve the higher levels of a system in energy management? One example is energy-aware battery management [88], [89], where the operating system identifies the desired system configurations dynamically in accordance with real-time load requirements and adaptively reconfigures the batteries to improve its energy efficiency. Many research questions follow:

- What managements can be performed by the higher levels of a system to save energy? For example, offloading more computation-intensive tasks to the remote edge device [47] for execution?
- How to apply dynamic voltage and frequency scaling techniques on the heterogeneous computing platform to save energy while the real-time requirements of the autonomous driving system are still maintained?
- Can edge devices in the digital driving circumstance be used to reduce energy demand on an autonomous vehicle? What is the range of possible approaches, and what are their relative merits [90]?

V. CONCLUSION

Autonomous driving will be a fruitful source of research problems in computer systems for many years to come. This paper presents challenges and opportunities for future research in connected and collaborative autonomous diving systems where traditional autonomous vehicles are running with other participants such as edge devices and the cloud. As a safety-critical and real-time heterogeneous platform, successful management of the computing tasks demands an effective coordination between different modules, from system design to downstream applications. The involved domains include but are not limited to real-time systems, sensor and

wireless networks, edge and cloud computing, and machine learning. The identified challenges lead to opportunities in each individual area, and hopefully shed light on an inter-disciplinary study for advancing autonomous driving.

REFERENCES

[1] Company history: Benz patent motor car: The first automobile. Https://www.daimler.com/company/tradition/company-history/1885-1886.html.

[2] Tesla roadster. Https://www.tesla.com/roadster.

[3] Apolong. Https://en.wikipedia.org/wiki/Apolong.

[4] Waymo. Https://en.wikipedia.org/wiki/Waymo.

[5] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989.

[6] Mcity. Https://mcity.umich.edu/.

[7] Hydraone. Http://thecarlab.org/hydraone/.

[8] C. Basaran and K.-D. Kang, "Supporting preemptive task executions and memory copies in GPGPUs," in *2012 24th Euromicro Conference on Real-Time Systems*. IEEE, 2012, pp. 287–296.

[9] G. A. Elliott and J. H. Anderson, "Globally scheduled real-time multiprocessor systems with GPUs," *Real-Time Systems*, vol. 48, no. 1, pp. 34–74, 2012.

[10] ——, "Robust real-time multiprocessor interrupt handling motivated by GPUs," in *2012 24th Euromicro Conference on Real-Time Systems*. IEEE, 2012, pp. 267–276.

[11] ——, "Exploring the multitude of real-time multi-GPU configurations," in *2014 IEEE Real-Time Systems Symposium*. IEEE, 2014, pp. 260–271.

[12] G. A. Elliott, B. C. Ward, and J. H. Anderson, "GPUSync: A framework for real-time GPU management," in *2013 IEEE 34th Real-Time Systems Symposium*.

[13] S. Kato, K. Lakshmanan, A. Kumar, M. Kelkar, Y. Ishikawa, and R. Rajkumar, "RGEM: A responsive GPGPU execution model for runtime engines," in *2011 IEEE 32nd Real-Time Systems Symposium*. IEEE, 2011, pp. 57–66.

[14] S. Kato, K. Lakshmanan, R. Rajkumar, and Y. Ishikawa, "TimeGraph: GPU scheduling for real-time multi-tasking environments," in *Proc. USENIX ATC*, 2011, pp. 17–30.

[15] S. Kato, M. McThrow, C. Maltzahn, and S. Brandt, "Gdev: First-class GPU resource management in the operating system," in *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)*, 2012, pp. 401–412.

[16] Z. Dong, C. Liu, S. Bateni, Z. Kong, L. He, L. Zhang, R. Prakash, and Y. Zhang, "A general analysis framework for soft real-time tasks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1222–1237, 2018.

[17] U. Verner, A. Mendelson, and A. Schuster, "Batch method for efficient resource sharing in real-time multi-GPU systems," in *International Conference on Distributed Computing and Networking*. Springer, 2014, pp. 347–362.

[18] U. Verner, A. Schuster, M. Silberstein, and A. Mendelson, "Scheduling processing of real-time data streams on heterogeneous multi-GPU systems," in *Proceedings of the 5th Annual International Systems and Storage Conference*. ACM, 2012, p. 8.

[19] H. Zhou, G. Tong, and C. Liu, "GPES: A preemptive execution system for GPGPU computing," in *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2015, pp. 87–97.

[20] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "GPU scheduling on the NVIDIA TX2: Hidden details revealed," in *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2017, pp. 104–115.

[21] M. Yang, N. Otterness, T. Amert, J. Bakita, J. H. Anderson, and F. D. Smith, "Avoiding pitfalls when using NVIDIA GPUs for real-time tasks in autonomous systems," in *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[22] K. Group, "OpenVX homepage," https://www.khronos.org/openvx/.

[23] G. A. Elliott, K. Yang, and J. H. Anderson, "Supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms," in *2015 IEEE Real-Time Systems Symposium*. IEEE, 2015, pp. 273–284.

[24] K. Yang, G. A. Elliott, and J. H. Anderson, "Analysis for supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms," in *Proceedings of the 23rd International Conference on Real Time and Networks Systems*. ACM, 2015, pp. 77–86.

[25] Z. Dong, C. Liu, A. Gatherer, L. McFearin, P. Yan, and J. H. Anderson, "Optimal dataflow scheduling on a heterogeneous multiprocessor with reduced response time bounds," in *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[26] K. Yang, M. Yang, and J. H. Anderson, "Reducing response-time bounds for dag-based task systems on heterogeneous multicore platforms," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. ACM, 2016, pp. 349–358.

[27] M. Yang, T. Amert, K. Yang, N. Otterness, J. H. Anderson, F. D. Smith, and S. Wang, "Making OpenVX really "real time"," in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2018, pp. 80–93.

[28] T. Amert, S. Voronov, and J. H. Anderson, "OpenVX and real-time certification: The troublesome history," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019.

[29] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *arXiv preprint arXiv:1704.05519*, 2017.

[30] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[31] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2011.

[32] L. Ladickỳ, C. Russell, P. Kohli, and P. H. Torr, "Associative hierarchical random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1056–1077, 2013.

[33] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr, "Graph cut based inference with co-occurrence statistics," in *European conference on computer vision*. Springer, 2010, pp. 239–253.

[34] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[35] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[36] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *CVPR 2011*. IEEE, 2011, pp. 1265–1272.

[37] A. Milan, K. Schindler, and S. Roth, "Detection-and trajectory-level exclusion in multiple object tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3682–3689.

[38] W. Choi, C. Pantofaru, and S. Savarese, "A general framework for tracking multiple people from a moving camera," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1577–1591, 2012.

[39] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *2008 IEEE Conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[40] R. B. Wynn, V. A. Huvenne, T. P. Le Bas, B. J. Murton, D. P. Connelly, B. J. Bett, H. A. Ruhl, K. J. Morris, J. Peakall, D. R. Parsons *et al.*, "Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience," *Marine Geology*, vol. 352, pp. 451–468, 2014.

[41] P. Katsigiannis, L. Misopolinos, V. Liakopoulos, T. K. Alexandridis, and G. Zalidis, "An autonomous multi-sensor uav system for reduced-input precision agriculture applications," in *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2016, pp. 60–64.

[42] L. Techy, D. G. Schmale III, and C. A. Woolsey, "Coordinated aerobiological sampling of a plant pathogen in the lower atmosphere using two autonomous unmanned aerial vehicles," *Journal of Field Robotics*, vol. 27, no. 3, pp. 335–343, 2010.

[43] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*. IEEE, 2007, pp. 239–243.

[44] A. Burns and R. Davis, "Mixed criticality systems-a review," *Department of Computer Science, University of York, Tech. Rep*, 2019, online at https://www-users.cs.york.ac.uk/burns/review.pdf.

[45] Tesla driver killed in crash with autopilot active, nhtsa investigating. Https://www.theverge.com/2016/6/30/12072408/tesla-autopilot-car-crash-death-autonomous-model-s.

[46] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[47] Z. Dong, Y. Liu, H. Zhou, X. Xiao, Y. Gu, L. Zhang, and C. Liu, "An energy-efficient offloading framework with predictable temporal correctness," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 19.

[48] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Transactions on computers*, vol. 39, no. 9, pp. 1175–1185, 1990.

[49] M.-I. Chen and K.-J. Lin, "Dynamic priority ceilings: A concurrency control protocol for real-time systems," *Real-Time Systems*, vol. 2, no. 4, pp. 325–346, 1990.

[50] T. P. Baker, "Stack-based scheduling of realtime processes," *Real-Time Systems*, vol. 3, no. 1, pp. 67–99, 1991.

[51] R. Rajkumar, L. Sha, and J. P. Lehoczky, "Real-time synchronization protocols for multiprocessors," in *Proceedings. Real-Time Systems Symposium*. IEEE, 1988, pp. 259–269.

[52] R. Rajkumar, "Real-time synchronization protocols for shared memory multiprocessors," in *Proceedings., 10th International Conference on Distributed Computing Systems*. IEEE, 1990, pp. 116–123.

[53] P. Gai, M. Di Natale, G. Lipari, A. Ferrari, C. Gabellini, and P. Marceca, "A comparison of mpcp and msrp when sharing resources in the janus multiple-processor on a chip platform," in *The 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings.* IEEE, 2003, pp. 189–198.

[54] A. Block, H. Leontyev, B. B. Brandenburg, and J. H. Anderson, "A flexible real-time locking protocol for multiprocessors," in *13th IEEE international conference on embedded and real-time computing systems and applications (RTCSA 2007)*. IEEE, 2007, pp. 47–56.

[55] B. B. Brandenburg and J. H. Anderson, "Optimality results for multiprocessor real-time locking," in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 49–60.

[56] ——, "The OMLP family of optimal multiprocessor real-time locking protocols," *Design automation for embedded systems*, vol. 17, no. 2, pp. 277–342, 2013.

[57] B. C. Ward and J. H. Anderson, "Supporting nested locking in multiprocessor real-time systems," in *2012 24th Euromicro Conference on Real-Time Systems*. IEEE, 2012, pp. 223–232.

[58] ——, "Fine-grained multiprocessor real-time locking with improved blocking," in *Proceedings of the 21st international conference on real-time networks and systems*. ACM, 2013, pp. 67–76.

[59] ——, "Multi-resource real-time reader/writer locks for multiprocessors," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 177–186.

[60] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[61] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton, "A proportional share resource allocation algorithm for real-time, time-shared systems," in *17th IEEE Real-Time Systems Symposium*. IEEE, 1996, pp. 288–299.

[62] L. Liu, Y. Yao, R. Wang, B. Wu, and W. Shi, "Equinox: A road-side edge computing experimental platform for cavs," in *2020 IEEE International Conference on Connected and Autonomous Driving (MetroCAD)*.

[63] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," in *Proceedings 21st IEEE Real-Time Systems Symposium*. IEEE, 2000, pp. 13–23.

[64] Z. Dong and C. Liu, "Closing the loop for the selective conversion approach: A utilization-based test for hard real-time suspending task systems," in *2016 IEEE Real-Time Systems Symposium (RTSS)*.

[65] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Real-Time Systems*, 2002.

[66] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.* IEEE, 2002, pp. 71–80.

[67] A. Block, J. H. Anderson, and G. Bishop, "Fine-grained task reweighting on multiprocessors," in *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*. IEEE, 2005, pp. 429–435.

[68] Z. Deng and J.-S. Liu, "Scheduling real-time applications in an open environment," in *Proceedings Real-Time Systems Symposium*. IEEE, 1997, pp. 308–319.

[69] A. K. Mok, X. Feng, and D. Chen, "Resource partition for real-time systems," in *Proceedings Seventh IEEE Real-Time Technology and Applications Symposium*. IEEE, 2001, pp. 75–84.

[70] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003.*.

[71] G. Lipari and E. Bini, "Resource partitioning among real-time applications," in *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.* IEEE, 2003, pp. 151–158.

[72] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using edp resource models," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*. IEEE, 2007, pp. 129–138.

[73] I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering of multiprocessors," in *2008 Euromicro Conference on Real-Time Systems*. IEEE, 2008, pp. 181–190.

[74] A. Easwaran, I. Shin, and I. Lee, "Optimal virtual cluster-based multiprocessor scheduling," *Real-Time Systems*, 2009.

[75] H. Leontyev and J. H. Anderson, "A hierarchical multiprocessor bandwidth reservation scheme with timing guarantees," *Real-Time Systems*, vol. 43, no. 1, pp. 60–92, 2009.

[76] E. Bini, G. Buttazzo, and M. Bertogna, "The multi supply function abstraction for multiprocessors," in *2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, 2009, pp. 294–302.

[77] E. Bini, M. Bertogna, and S. Baruah, "Virtual multiprocessor platforms: Specification and use," in *2009 30th IEEE Real-Time Systems Symposium*. IEEE, 2009, pp. 437–446.

[78] G. Lipari and E. Bini, "A framework for hierarchical scheduling on multiprocessors: from application requirements to run-time allocation," in *2010 31st IEEE Real-Time Systems Symposium*.

[79] M. Xu, L. T. X. Phan, O. Sokolsky, S. Xi, C. Lu, C. Gill, and I. Lee, "Cache-aware compositional analysis of real-time multicore virtualization platforms," *Real-Time Systems*, 2015.

[80] K. Yang and J. H. Anderson, "On the dominance of minimum-parallelism multiprocessor supply," in *2016 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2016, pp. 215–226.

[81] Tesla's latest autopilot death looks just like a prior crash. Https://www.wired.com/story/teslas-latest-autopilot-death-looks-like-prior-crash/.

[82] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, 2017.

[83] Y. Wang, W. Chen, W. Zheng, H. Huang, W. Zhang, and H. Liu, "Tracking hit-and-run vehicle with sparse video surveillance cameras and mobile taxicabs," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 495–504.

[84] H. G. Seif and X. Hu, "Autonomous driving in the icity—hd maps as a key challenge of the automotive industry," *Engineering*, vol. 2, no. 2, pp. 159–162, 2016.

[85] Z. Dong and C. Liu, "Analysis techniques for supporting hard real-time sporadic gang task systems," *Real-Time Systems*, vol. 55, no. 3, pp. 641–666, 2019.

[86] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[87] S. Baruah, "Rapid routing with guaranteed delay bounds," in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2018, pp. 13–22.

[88] L. He, L. Gu, L. Kong, Y. Gu, C. Liu, and T. He, "Exploring adaptive reconfiguration to optimize energy efficiency in large-scale battery systems," in *2013 IEEE 34th Real-Time Systems Symposium*. IEEE.

[89] Z. Dong, C. Liu, Y. Li, J. Bao, Y. Gu, and T. He, "Rec: Predictable charging scheduling for electric taxi fleets," in *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2017, pp. 287–296.

[90] L. Liu, J. Chen, M. Brocanelli, and W. Shi, "E2m: an energy-efficient middleware for computer vision applications on autonomous mobile robots," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. ACM, 2019, pp. 59–73.