VECFrame: A Vehicular Edge Computing Framework for Connected Autonomous Vehicles

Sihai Tang*, Bruce (Haidi) Chen*, Harold Iwen†, Jason Hirsch*, Song Fu*, Qing Yang*,
Paparao Palacharla‡, Nannan Wang‡, Xi Wang‡, and Weisong Shi§
*Department of Computer Science and Engineering, University of North Texas
Email: {SihaiTang, HaidiChen, JasonHirsch}@my.unt.edu; {Song.Fu, Qing.Yang}@unt.edu
†Department of Computer Science, University of Minnesota Duluth, Email: iwenx014@d.umn.edu
‡Fujitsu Network Communications, Email: {Paparao.Palacharla, Nannan.Wang, Xi.Wang}@fujitsu.com
§Department of Computer Science, Wayne State University, Email: weisong@wayne.edu

***Abstract*—Autonomous vehicle systems require sensor data to make crucial driving and traffic management decisions. Reliable data as well as computational resources become critical. In this paper, we develop a Vehicular Edge Computing FRAMEwork (VECFrame) for connected and autonomous vehicles (CAVs) exploring containerization, indirect communication, and edge-enabled cooperative object detection. Through our framework, the data, generated by on-board sensors, is used towards various edge serviceable tasks. Due to the limited view of a vehicle, sensor data from one vehicle cannot be used to perceive road and traffic condition of a larger area. To address this problem, VECFrame facilitates data transfer and fusion and cooperative object detection from multiple vehicles. Through real-world experiments, we evaluate the performance and robustness of our framework on different device architectures and under different scenarios. We demonstrate that our framework achieves a more accurate perception of traffic condition via vehicle-edge data transfer and on-edge computation.**

***Index Terms*—Edge computing, Connected autonomous vehicles, Vehicle-edge system, Cooperative computing, Performance analysis.**

I. Introduction

Autonomous vehicles are becoming more and more relevant in today's society. There is no denying the benefits of an autonomous vehicle in terms of driver and pedestrian safety. By delegating the driving decision to on-board computing units, the driver-related issues such as driving under the influence and other human operating errors are significantly reduced. In addition to the safety benefits, we are also witnessing a blossom of various other uses [25], [40], [32], with amber alert detection and collusion avoidance being the most prominent [41].

To facilitate the self-driving process, various sensors need to relay their sensing data of the surrounding environment to the on-board computing unit. This is usually handled by an array of sensors such as the LiDAR, cameras, radar, GPS, IMU, and more. Due to the vast array of sensors, it is estimated that an autonomous vehicle will generate 4 terabytes of data or more in two hours [2]. To enhance driving, connected and autonomous vehicle (CAV) technology enables raw-data level and feature-map level data sharing among vehicles [10], [11], [6], which utilizes extraneous data from other vehicles to drastically improve the detection capabilities of a single vehicle.

Currently, most car manufacturers focus on uploading their data to the cloud. However, the expensive data transmission, exacerbated network congestion and prolonged latency between vehicles and the cloud make real-time object detection inaccurate if not infeasible.

In this paper, we contribute our system, vehicular edge computing framework (VECFrame), for the transfer and fusion of images between vehicles and edge nodes to achieve cooperative perception. We test the use of edge nodes as a targeted and purposed system to facilitate the exchange of large sensor data towards safer driving. VECFrame utilizes the benefits of the close proximity of vehicles to edge nodes to achieve direct and unfettered data dissemination and usage, thereby allowing for a variety of use cases. We introduce an efficient approach using modular containers in a framework to disseminate data between vehicles and edge nodes, and achieve a scalable framework towards cooperative perception tasks through facilitating the flow of data. Our VECFrame system opens up the data for more immediate processing on edge nodes closer to vehicles. As the current bottleneck for data exists in the limited means of communication, we incorporate the power of edge with our system, mitigating the bottleneck through the edge. We develop VECFrame through containerization so it is scalable and platform independent. To ensure that VECFrame can handle real-world communication and computation loads, we rigorously test our VECFrame in our experimentation.

We have implemented a prototype of VECFrame and evaluated its performance on a real CAV-edge test platform. The experimental results show the developed VECFrame achieves a horizontal scaling with more hardware utilized by VECFrame. VECFrame can successfully handle cooperative object detection in a scalable fashion. Experiments also show that VECFrame is capable of disseminating data in a facilitated manner through the edge servers with an increase of 40% to 350% in throughput with the edge as opposed to without.

The rest of the paper is organized as follows. We introduce related works in Section II. We outline the design principals in Section III and describe the details of VECFrame in Section

2021 IEEE International Conference on Edge Computing (EDGE) | 978-1-6654-0062-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/EDGE53862.2021.00019

2767-9918/21/$31.00 ©2021 IEEE
DOI 10.1109/EDGE53862.2021.00019

IV. We show the computational workflow in Section V and detail our experimentation and results in Section VI. Finally, we conclude our paper and discuss future works in Section VII.

II. Related Works

In 2014, the Society of Automotive Engineers(SAE) international announced in at the Taxonomy of automated driving technology the SAE J3016 standard that helps define levels of autonomy in vehicles[5].

In the J3016 standard, the process of going from full manual to full automation goes through 6 main stages, with each stage highlighting a level of automation. However, in the development process of autonomous vehicles, we encounter two paths, intelligence and networking. Just as a human, the autonomous vehicle must have both intelligence and networking to fully succeed the human in the task of full self driving.

Modern vehicles have access to radio and cellular technologies that allow for constant connectivity [37]. The communication by the vehicles leads to the realization of Vehicle to X (vehicles, infrastructures, roads, people, cloud, etc.). Autonomous vehicles goes a step beyond these by adding powerful onboard sensors and computational devices.

Vehicular communication through DSRC [22] and cellular data are the two widely used technologies. With 5G in deployment across the world, we see even more avenues being opened up [42]. A recent work [42] studies the advantages and limitations of these technologies from both technical and non-technical perspectives.

With the public adoption of autonomous vehicles, all vehicles, both autonomous ones and human operated ones, will be able to benefit from the information that autonomous vehicles can provide, should communication between the former and latter become possible [42]. In [19], challenges are presented as the widespread deployment of autonomous vehicles will impact multiple sectors of society.

Going further in this direction, [33] develops a Vehicle to X framework that studies the collaboration between devices, edge servers, and cloud services. Similarly, Liu et al. analyze the interaction of the Human and vehicle under the modern understanding of driving assistance as well as autonomous features [26].

In works such as [23] and[9], the potential of IoT and Edge are discussed. Tasks such as low latency demand as well as distributed computing as easily achieved through edge devices. Tasks such as deep learning can be approached by the edge in a distributed computing method [9], [17]. Issues like scheduling, power consumption and security in edge computing are studied in [12], [39], and [36].

A pioneering work on vehicular edge computing for video crowd sourcing is presented in [43]. Here, Zhu et al. present a novel method of turning public transportation mediums such as busses and taxis into vehicular fog nodes. With this approach, the edge computation is able to reach many more vehicles than through stationary service points. However, in [43], they also point out the deficiencies of using direct video transmission.

Recently, Franke et al. address the barriers and challenges that autonomous vehicles face in urban traffic [13]. Additionally, work such as [34] introduce and implement low power systems targeting OBU units in autonomous vehicles. These works highlight the issues facing autonomous vehicles when relying only on the enclosed system without edge.

Additional research into how Edge can be dynamically optimized, based on the algorithm used or based on the use case, are examined in works such as [28], [27], where Luo et al. propose an algorithm based approach to adapt the scheduling of the edge based on priority. Luo et al. also emphasize the constraints faced by the limited resources a edge node has available when tasked by multiple vehicles.

III. Design Principles of the CAV-Edge Framework

With a better understanding of the barriers and challenges of using edge computing for CAVs, we design **VECFrame**, a Connected Autonomous Vehicle-Edge framework, to support lightweight and agile data processing for CAVs. To manifest the advantages of edge computing, we adhere to the following ideologies while designing our framework. VECFrame must be *data-friendly*, *adaptable*, and *scalable*. (1) VECFrame is designed for facilitating sensor data transmission and processing, therefore, it must be data friendly and can handle huge amount of data transferred from vehicles to edge nodes and processed on the edge. (2) VECFrame must also be adaptable, in terms of running various types of applications written in different program languages and executed on heterogeneous hardware. (3) Last but not least, VECFrame needs to be scalable; There is likely a large number of vehicles at a location of a reasonable service zone. The performance of edge communication and computation should scale well to accommodate the increased number of requests and amount of data.

A. Efficient Data Dissemination

V2X communications of autonomous vehicles have been a hot research topic, e.g., the PROMETHEUS and PATH projects in Europe and United States, respectively [21], [7]. The recent advent of cooperative perception techniques [10] require far more data than what traditional V2X can support.

With the existing V2X techniques, the information is broadcast to all nearby vehicles or unicast to specific vehicles. It is not clear, however, what information is sent to which vehicles, leading to an inefficient data dissemination for CAVs. This problem becomes even worse when massive amount of sensor or feature map data need to be transmitted between vehicles and infrastructures [10], [6].

To make good use of the limited network bandwidth available for vehicular communications, it is critical to design an efficient data dissemination mechanism for CAVs. To this end, we adopt the publish-subscribe paradigm [20] in VECFrame, which supports efficient information dissemination between vehicles and edge nodes. Specifically, we make use of the Message Queuing Telemetry Transport (MQTT) [18] and Wget protocols [4] to achieve efficient data dissemination for CAVs.

B. Adaptability

To ensure our VECFrame framework is capable of handling a wide variety of use cases and workloads, we need to make it adaptable so that different algorithms/programs can be packaged into individual modules which are then deployed in a flexible manner. In the current day workflow, a myriad of different languages are used for different programs, and this has both positive and negative effects. For autonomous vehicles, there is no set of defined guidelines that all manufacturers follow. Take Nvidia PX2, the OBU for our CAV enabled vehicle, for example, while the processing unit is powerful, countless issues arise when custom code is ran on the hardware. More modern versions of this type of OBU, such as the Drive PX Pegasus or Drive AGX [3], are more open and support more customization, but they are still packaged systems with constraints. Different operating systems, chipset architectures, software versions and other constraints tie down and restrict the usefulness of a tailored program. To address these issues, we explore containerization techniques in VECFrame. Containers are lightweight and agile for running modular edge workloads, due to the fact that they share the same kernel [38].

C. Scalability

As time progresses, we will start to see more and more autonomous vehicles on the roads. This increase will require VECFrame to be scalable in order to accommodate the growth. There are two methods of upscaling a system: vertical and horizontal [16]. Vertical scaling entails an increase in hardware performance while horizontal scaling entails an increase in the number of devices. In this paper, we concentrate on horizontal scaling, leaving the vertical scaling as an issue for future research.

With the increase in vehicles, wide-spread data transmission would cause the vehicle on-board unit (OBU) to become the bottleneck of data dissemination, due to its limited resources, taxing the resources needed for reliable data processing. To address this bottleneck problem, we leverage edge devices/nodes to handle scheduling and deep learning-based processing of vehicular data. The scalability of edge nodes can be tackled by either scale-up (i.e., using more powerful edge nodes) and/or scale-out (i.e., using more edge nodes). It is cost and energy efficient to achieve horizontal scaling by means of containerization and edge computing.

IV. VECFrame: A CAV-Edge Communication and Computation Framework

Between manufacturers and car models, there are different hardware and software in the available and upcoming autonomous cars. Adding to the fact that we need to account for a scalable solution for our framework, we consider using containers in our VECFrame framework.

As Docker is an industry standard for containerization, rapid development and deployment of programs, and it is built to accommodate horizontal scaling through the use of kubernetes, swarm, or other solutions that supports docker containers [8].

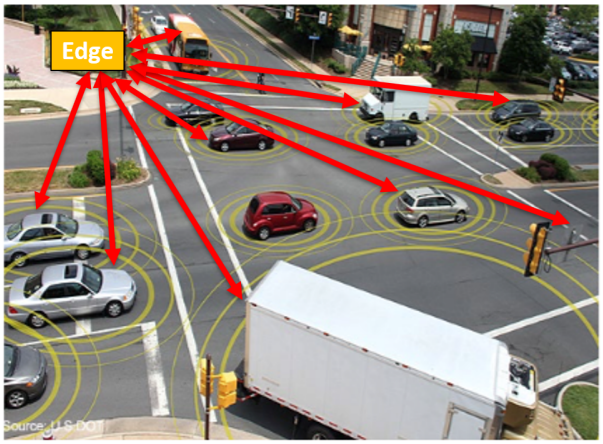


Fig. 1: An edge-centric communication and computation system is formed in the scenario where roadside edge node(s) facilitates data dissemination and processing to reduce congestion and delay.

It is for this reason that we decide to build VECFrame with modular containers rather than an single purpose and platform bound program.

A. VECFrame Communication Subsystem

The on-board units of autonomous vehicles leverage the computational resources available on edge nodes to perform computational tasks.

Edge nodes are owned by edge service providers, e.g., telecom companies or state departments of transportation. Car manufacturers setup service agreements with the edge service providers, as well as obtain the required mechanisms to coordinate with roadside edge nodes. Moreover, the security and privacy protection measures are entailed through the service agreement and are tied in with the handshaking and discovery process.

A vehicle can discover edge nodes, and their provided services, through a service discovery process. Depending on the edge service providers, the service discovery processes might not be necessarily the same. Through such services, a vehicle learns the security mechanisms adopted by the edge nodes, computational resources available for request, service APIs, etc. Based on the obtained information, a handshake process is initiated between a vehicle and an edge node to ensure the vehicle is connected to the edge.

After a connection is built, the vehicle will coordinate its current tasks with the edge node, e.g., offloading certain tasks to the edge. The edge node, on the other hand, could retrieve data collected by the vehicle or multiple vehicles to accomplish its own task. As data from other vehicles could be useful and/or needed by the edge node(s) to perform tasks such as cooperative perception, a collective communication mechanism is needed in VECFrame. For an effective collection/dissemination of data for different tasks (topics), we exploit the publish-subscribe paradigm in VECFrame's communication subsystem.

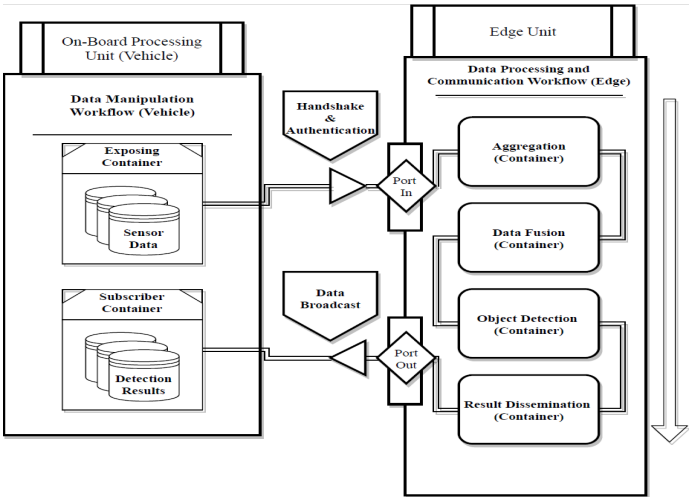


Fig. 2: Vehicles and edge nodes run VECFrame container based workflows.

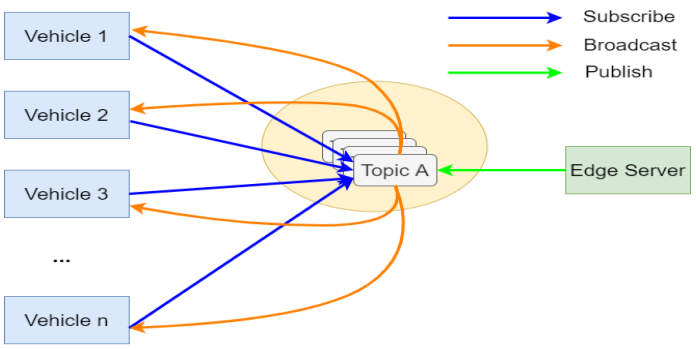


Fig. 3: Task facilitation and result dissemination using a publish-subscribe paradigm.

One major benefit of VECFrame is that the computational results obtained on the edge nodes can be broadcast to the associated vehicles, saving computational resources on these vehicles. Most importantly, several tasks such as obtaining real-time mapping information must be done through the edge servers, due to the limited field of views on individual vehicles. Fig. 1 shows an edge-centric communication system where vehicles, traffic infrastructure (e.g., smart cameras on traffic lights), and other smart objects/devices send/receive data to/from edge nodes collectively or individually to perform CAV and smart transportation related tasks. Because the edge becomes not only the data processor but also the data facilitator, we anticipate an overall reduction in network traffic and congestion. Without the introduction of edge nodes, data needs to be transmitted multiple times to multiple destinations. This is not only redundant but also costly, which is examined in Section VI.

There are distinctive advantages of this design, e.g., fostering interoperability is one of them. Traditionally, vehicles made by different manufactures cannot communicate with each other, even with the DSRC standard being employed as the main protocol for vehicular communications. Through the edge nodes, however, different vehicles can share their data between each other and with the traffic infrastructure. Besides interoperability, efficient data dissemination can be achieved; only the relevant vehicles which are interested or involved in certain tasks will participate in data exchange, which makes both network communication and edge computing efficient. This is achieved through a publish-subscribe paradigm employed in the framework, which is detailed in Section V-A. In addition, the edge nodes provide an extra layer of security and privacy protection to vehicles, which either share their sensor data to others or consume the data provided by the local edge without going through the public Internet or possible leakage in the cloud. With this in mind, VECFrame opens up opportunities for more secure applications to be developed and deployed on CAVs.

V. VECFrame Computation Workflow

Starting with the task type, we know from works such as [40], [41] that edge servers can service anything from emergencies to infotainment. With tasks ranging from time critical to mundane entertainment, an edge server needs to prioritize its service schedule. Before a task is accepted, an edge server evaluates if there are sufficient computational resources that are suited for the requested task. This is achieved in the handshaking process in which a vehicle requesting edge computing service needs to communicate its task as well as its credentials as part of the handshake process. Based on the specific needs, an edge node either accepts or denies the vehicle's requests.

When a requesting task is accepted, the edge node converts the request into multiple modules to support parallel processing and reliable computing. As seen in Fig. 2, the vehicle has two lightweight components (as containers) that handle data exposure and data reception separately. This separation allows for enhanced data isolation, i.e., only the data allowed for sharing is exposed to an edge node. The sensor data of a vehicle is dynamic, and with communication and task priority in mind, VECFrame distincts data exposure versus data uploading. Exposing data and uploading data are two different concepts in our framework. Exposing data makes the data available so that an edge node handles its scheduling. Whereas data uploading prepares the data at the time of upload, requiring precious OBU resources.

After data has been received by the edge node, it passes the received data to the appropriate workflow components (as containers) for processing. For example, the data received from multiple vehicles or the traffic infrastructure is aggregated first to ensure all sensor data are placed into the same coordinate system. To achieve this goals, the location information of contributing vehicles (or roadside infrastructures), and the configuration and calibration details of these sensors are needed by the edge nodes. Aggregated data is then be fed into the fusion module, which effectively fuses the data generated from multiple vehicles. Different data fusion algorithms might be applied in this module, for example, a feature map

based fusion solution could be appropriate for LiDAR data fusion [10]. With the fused data, an object detection module is executed to discover the locations and types of objects in the fused data. Again, depending on the nature of the fused data and the edge node's current load, an object detection algorithm is performed. There are many 2D and 3D object detection algorithms available for CAV applications, e.g., those listed on the KITTI website [15]. VECFrame provides an open framework where different object detection algorithms can be employed and evaluated. Finally, the object detection results based on the fused data are then made available to all vehicles that subscribed to this information (i.e., topic). The dissemination module efficiently multicasts the obtained object detection results to the interested vehicles.

The entire data processing is decoupled into multiple modules, i.e., data aggregation, data fusion, object detection, and results multicast, which improves the system's robustness, adaptability, and interoperability. More importantly, the algorithms (e.g., data fusion, object detection, and multicasting) employed in different modules can vary to accommodate different CAV applications and available resources on edge nodes.

A. Effective CAV-Edge Communication

Efficient data communication is important in VECFrame, due to the sheer amount of data that needs to be exchanged between vehicles and edge nodes. It is not affordable if the sensor data is transmitted multiple times or re-transmitted due to security issues. As studied by Intel, an autonomous vehicle equipped with high end sensors such as LiDAR can generate around 4 terabytes of data in two hours [31], [2]. In addition to the sheer quantity of data, we also need to consider the integrity of the data after transmission.

In Fig. 2, data generation (vehicles) and consumption (edge nodes) are separated from each other. In VECFrame's publish-subscribe subsystem, a broker (which can reside on an edge node) maintains a list of topics and vehicles (as data generators) publish their sensor data via the broker. Through the usage of this subsystem, the edge can effectively handle multiple different types of workflow through piping the needed data from the necessary containers. This process reduces the workload. Both vehicles and edge nodes subscribe certain topics maintained by the broker. When new data arrives, the broker notifies the subscribers in various manners, based on the characteristics of the data. The broker can be deployed in a distributed manner, e.g., on multiple edge nodes, to achieve a better scalability and reliability.

From this, we exploit the MQ Telemetry Transport (MQTT) [18] as the result dissemination protocol in VECFrame. MQTT is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks.

Since the raw data costing more bandwidth, efficient and reliable transmission is critical. We have tested a myriad of communication protocols as seen in Fig. 4 for both throughput and reliability. The results show that Wget and aria2cx outperform other data transfer methods in terms of speed and reliability. They are the better choice as the raw data dissemination protocol.

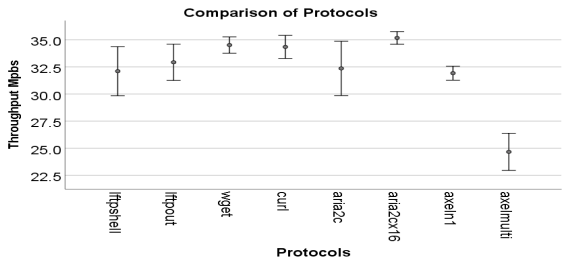


Fig. 4: Comparison of data transfer methods in terms of speed and reliability.

B. Edge Computing for CAVs

Edge nodes are close in proximity and thus low in latency, so we deem them as devices that provide extra computational resources. These possibilities range anywhere from real-time amber alert detection to infotainment for autonomous vehicles [32], [24].

Cooperative perception [10] can make use of our CAV-Edge framework to acquire the needed data and be able to service multiple vehicles that cooperate to extend the perception range and object detection accuracy. In the case of early fusion, fusion of raw data or feature data, the vehicle is required to pipeline that data to other vehicles while processing said data at the same time for itself. In works such as [6], early fusion plays an important factor to the vehicles in the region by significantly improving detection results from 30% to 95%.

However, the trade between extra improvements in detection versus taking up crucial on board computing resources is now the main issue. As fusion between different vehicles all require both location, orientation, as well as a myriad of other factors to function properly, the process of sending this data alone is taxing.

This does not stop at data fusion. While in a testing environment, we are able to run experiments according to the hardware available, the compiled software for the right platform, as well as use compatible data formatting that will not conflict. However, in the real world, such ideal conditions do not exist due to the lack of an uniform standard for CAV.

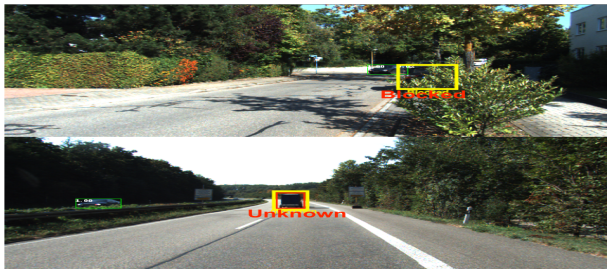


Fig. 5: Views of a real-world Detection error based on single vehicle

Fig. 6: Our CAV-Edge test platform is a Polaris GEM equipped with various sensors and onboard computing units.

Fig. 7: Views of a real-world parking lot from individual vehicles.

With that in mind, the usage of Edge to facilitate these endeavors become evident as the best option. With the Edge serving as the main point of data traffic, both the issues of stressing the on board unit as well as heterogeneous hardware and data formats can be resolved.

To show the limits of autonomous vehicles without edge support, we depict the results from a single vehicle in Fig. 5. In Fig. 5, we can clearly see that the top scenario has obstruction of a vehicle by bushes, while the bottom scenario failed to detect a vehicle with a trailer attachment. In these situations, an edge device running cooperative perception services can notify and warn the vehicle of such objects that the OBU has failed to detect. It is due to these faults that we decide to focus on prioritizing cooperative perception on the edge rather than other operational services for VECFrame.

In this paper, we study use cases related to 2D object detection of the traffic conditions.

VI. Performance Evaluation

We have implemented a prototype of VECFrame and evaluated its performance on a connected autonomous vehicle (CAV) test platform. Our CAV platform includes electric cars equipped with a variety of sensors, an onboard (on each electric car) computing unit, multiple roadside edge computing nodes, and on-campus edge servers. The sensors mounted on the electric cars are Velodyne Ultra PUCK VLP-16 LiDAR sensor, Delphi SRR2 radar sensor, Sekonix 120/60 degree field of view (FOV) cameras, and Xsens MTi-G710 GPS sensor. The onboard computing unit is NVIDIA DRIVE PX2. The edge nodes and servers are HPE ProLiant servers.

We used the following equipment in our framework experiment:

- Nvidia PX2
- 4K Capable 12 MegaPixel Camera
- Netgear r6020 Router
- 8xDell Optiplex 3060 Small Formfactor
- TP-link T2U wireless transceiver (DSRC Equivalent throughput - empirically tested)
- Snapdragon 845 SOC with 802.11 ac dual band Murata KM8509176 module
- 2xraspberry pi

Fig. 8: Multi-vehicle cooperative object detection results from the edge show a much wider perception range and an accurate detection of vehicles and pedestrians.

We design our experiments into two categories: tests of the VECFrame communication subsystem and tests of the edge computing subsystem. For the first stage of our experimentation, we empirically test a variety of data aggregation protocols in terms of their throughput and variance. Fig. 4 shows the measured throughput of eight data transfer tools. In the figure, we can see wget and aria2cx show a consistent throughput. While Wget shows its capability as a reliable and fast protocol, it does not come close to protocols, such as MQTT which is a lightweight, publish-subscribe protocol designed for mobile devices.

Next, we conduct experiments with a cooperative 2D object detection road monitoring as a use case. The experiments are performed in a controlled DSRC environment with eight vehicles (OBUs with our CAV test platform data) and one edge node. The data was gathered with the vehicle shown in Fig. 6.

A. Performance with Edge

The addition of edge nodes facilitates data transfer and processing. We conduct experiments, in a controlled DSRC environment, to fully evaluate all facets of VECFrame. We test framework configurations with and without an edge node. The results are shown in Fig. 9. On the left, we see that without the use of edge, the average throughput is 0.6 Mbps for MQTT and a mere 0.2 Mbps for Wget. However, with the presence of edge (in the figure on the right), we see a definite improvement

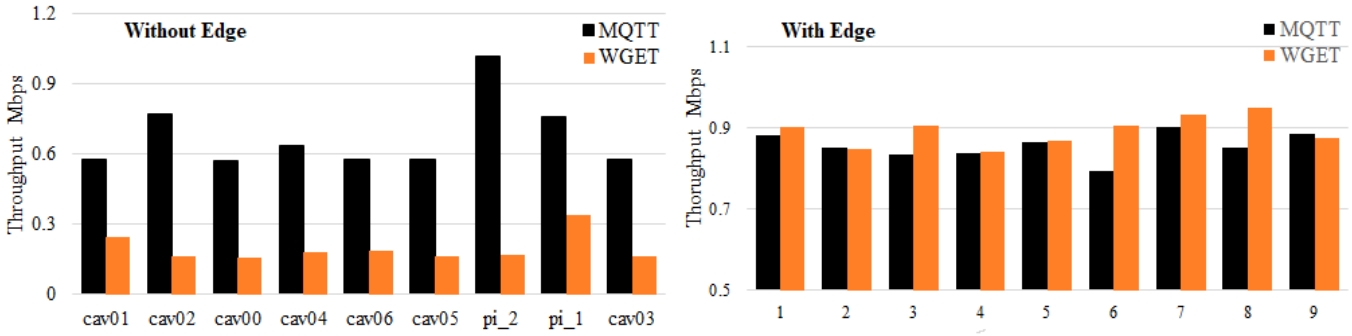


Fig. 9: Performance impact from edge computing. On the left, we see that without the use of an Edge node, the throughput is averaging 0.6 Mbps for MQTT and a mere 0.2 Mbps for Wget. However, with the presence of Edge node on the right, we see improvement for both MQTT (0.85 Mbps) and Wget (0.9 Mbps).

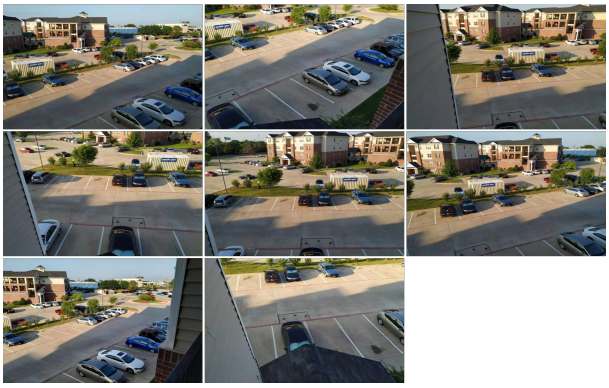


Fig. 10: We test the traffic cameras from stoplights in this scene. Individual images are shown.

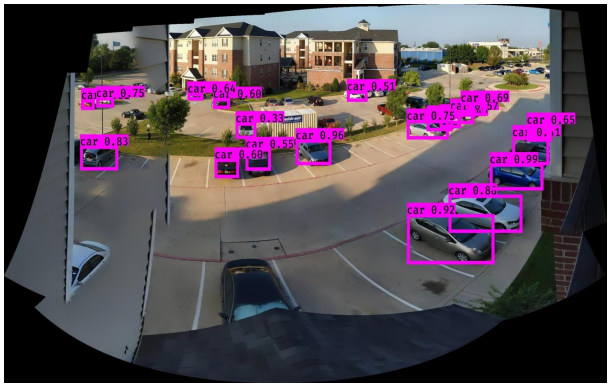


Fig. 11: Most of the vehicles are correctly detected, located, and counted in the vehicle-infrastructure cooperative perception using edge computing.

for both MQTT and Wget, averaging 0.85 Mbps (i.e., a 41.7% increase) and 0.9 Mbps (i.e., a 350% increase) respectively.

B. Multi-Vehicle and Vehicle-infrastructure Cooperative Object Detection Performance Evaluation

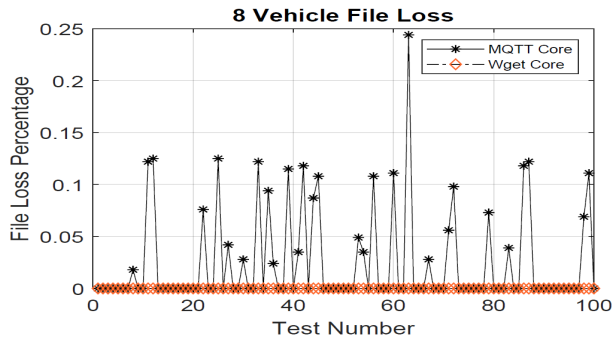
We design two sets of experiments for road and traffic monitoring. Peer vehicles share their 2D images in the first scenarios, and images from the transportation infrastructure (for example, cameras on traffic lights) are used in the second scenario.

Fig. 7 shows the images taken by individual vehicles of a real-world parking lot. The edge node takes multiple 2D images from vehicles through fusion and then object detection with OpenCV [29]and YOLOV3 [30] respectively in our framework. For an incoming vehicle, the results from the framework provide more than ample warning as seen in Fig. 8. In the results, we can clearly see a pedestrian on the far right hand side, which is well out of line of sight to vehicles approaching the intersection. Similar to the stationary sensors mentioned in [6], the edge node processes images taken and shared by the transportation infrastructure, such as traffic light cameras and roadside cameras, to help assist with road and traffic monitoring.

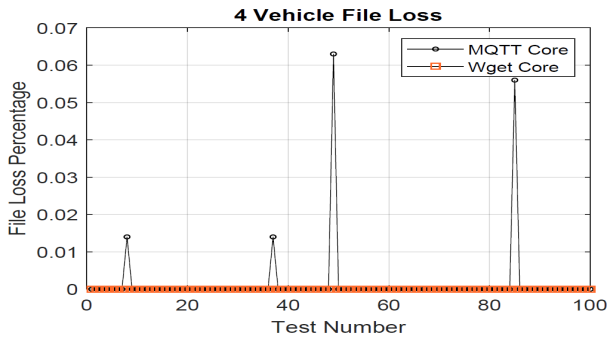
In our second set of experiments, we use a vantage point and angle akin to that of a traffic light camera as seen in Fig. 10. With this vantage point, we expect VECFrame helps facilitate the results of detection for incoming vehicles to be aware of the traffic and avoid accidents, and help regulate traffic management in accordance to the real-time condition.

Fig. 11 presents the experimental results for the traffic light scenario. Although there is some distortion due to image alignment in fusion, the combined view (perception range) is significantly enhanced and the vehicles are located, identified, and counted in a higher accuracy. With the edge leveraging data from the traffic infrastructure, we are disseminating potentially critical information to incoming vehicles. Factors such as sensor failure or environmental conditions might render an autonomous vehicle blind, but with vehicle-infrastructure cooperative perception using edge computing, we can potentially avoid these deadly accidents [35][14].

In addition to road and traffic monitoring, we have tested many other scenarios. The results are listed in in Table I with breakdown of the execution time of the four steps: data aggregation, data fusion, object detection on the fused data, and results dissemination. Each test is coincident with a different test



(a) 8 Vehicle File Loss



(b) 4 Vehicle File Loss

Fig. 12: Sets of 2, 4, and 8 vehicles are tested with the same payload. Reliability and data integrity (data loss) is evaluated.

scenario, with tests 1 through 11 detailing road intersections from various locations and weather conditions. We note that aside from the communication subsystem, VECFrame stalls on image fusion and detection due to hardware constraints (Our edge node has a quad-core i5 Intel processor). However, aside from hardware and optimization limitations, the vehicles are receiving the results with minimal latency from the edge.

Group	Agg.	Fusion	Detect	Multicast	Total (s)
Test 1	0.6	2.64696	3.63507	0.00544	6.88747
Test 2	0.3	1.60079	3.56692	0.00523	5.47294
Test 3	0.3	0.97034	3.57243	0.00461	4.84738
Test 4	0.2	1.72102	N/A	N/A	1.92102
Test 5	0.3	1.25113	3.61586	0.00539	5.17238
Test 6	0.3	1.72399	3.63339	0.00693	5.66431
Test 7	0.3	1.53299	3.63738	0.00671	5.47708
Test 8	0.3	1.89981	3.61449	0.00449	5.81879
Test 9	0.6	3.70769	3.68509	0.00872	8.0015
Test 10	0.3	2.01195	3.65668	0.00649	5.97512
Test 11	0.4	1.3641	3.61012	0.0046	5.37882

TABLE I: Breakdown of the execution time in the edge computing workflow.

C. Evaluation of Data Aggregation

We test MQTT and Wget for the performance of data aggregation and data integrity. MQTT and Wget play different roles in VECFrame. While MQTT is good at broadcasting small packets, Wget is more reliable in terms of data integrity. In this set of experiments, we evaluate vehicle-vehicle as well as vehicle-edge data aggregation.

1) Throughput and Scalability Evaluation: Scalability is an important aspect to the CAV-edge system. As our framework is containerized with docker and modular, horizontal scaling is achieved through the use of an efficient container orchestration system. Moreover, as VECFrame is modular, horizontal scaling requires that all of its components be capable of scaling. Computational components such as object detection and sensor data fusion can be scalable due to the increase in computational resource on edge nodes (scale-out and/or scale-up). However, communication may be bottlenecked due to the low-bandwidth wireless connections between vehicles and edge nodes and the large amount of sensor data (e.g., HD images) exchanged.

To evaluate the scalability of VECFrame's communication subsystem, we measure the its performance with an increased number of vehicles. We set up the vehicles in groups of 2, 4, and 8 with each vehicle generating a payload of 36 images. We then test each group over 100 runs by having all vehicles request the edge service simultaneously.

Fig. 14 shows the experimental results, i.e., data aggregation throughput. From the figure, we can clearly differentiate between Wget and MQTT, as Wget maintains a stable throughput rate of 20 Mbps for 2 vehicles, MQTT suffers Quality of Service(QoS) restrictions and only averages 3.9 Mbps. If we examine Fig. 12, we see a similar issue with MQTT versus Wget, where MQTT suffers from substancial data integrity issues with more vehicles than Wget, which suffered no data integrity loss. We further analyze the relationship between the increase in vehicles and the execution time. The results in Fig. 13 show a linear scalability is achieved for both with a R^2 value of 0.99 each, indicating a tight fit to the regression. We also note that both methods display an marked increase in variance as the number of vehicles increases. This is caused by contention on the edge node to aggregate more data as more vehicles are involved.

2) Reliability and Data Integrity Evaluation: As one of the critical functionalities in VECFrame, we need to ensure that MQTT and Wget are optimized for our framework. To this end, we need Wget to aggregate data from vehicles to the edge and MQTT to disseminate results from the edge back to vehicles. We have conducted 100 trials of the same workload in VECFrame. The trials are conducted in two setups, i.e., one group with a total payload of 36 images, and the other group with a payload of four images. Both groups are tested for 100 runs separately.

For data integrity, we need to check the percentage of data

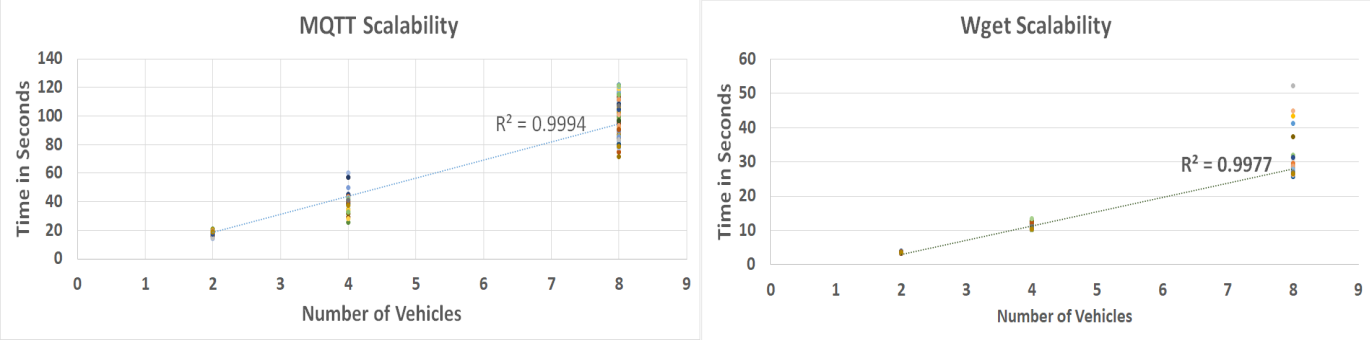
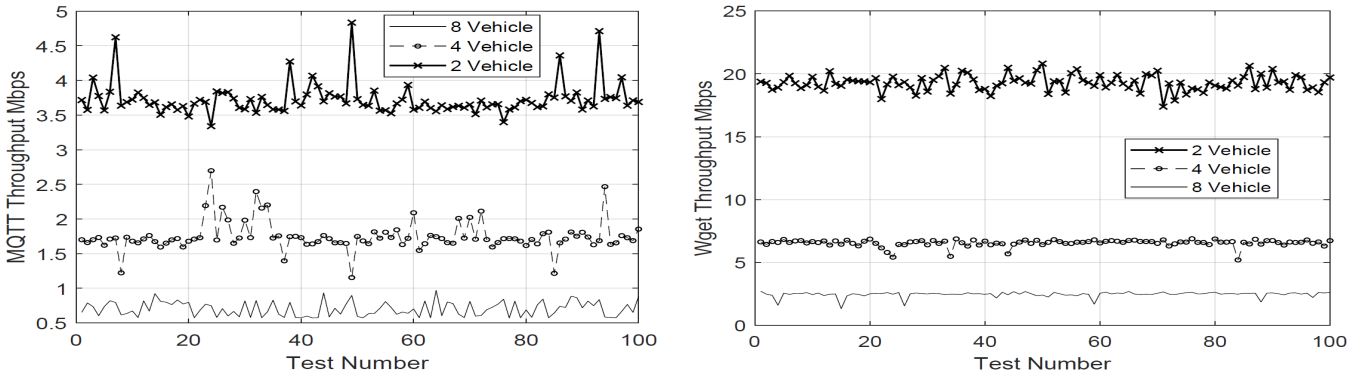


Fig. 13: Scalability of data aggregation. Linear regression fits both sets of data with a R^2 of 0.99.



(a) MQTT Throughput

(b) Wget Throughput

Fig. 14: Sets of 2, 4, and 8 vehicles are tested with the same payload. Performance (throughput) is evaluated.

loss that has occurred. As shown in Fig. 12, we have the comparison between MQTT and Wget from the experiment. As the file loss for our test with only two vehicles involved saw no data integrity loss, we did not include that in our figure. Due to the payload being images, we used a combination of mean squared error(MSE) and structural similarity index metric(SSIM) to check for image corruption on top of raw binary values,. However, as all of our results came back with negligible results, indicating that the loss of integrity does not impact the inference process, they were not included with the figure.

Fig. 12 plots the loss rate using MQTT and Wget, and from the figure, we see a vast differentiation between the two methods. Wget, despite being heavier as a protocol when compared to MQTT, performs significantly better in terms of reliability and data integrity protection; both are critical to data processing. As we can see, Wget was able to maintain complete data integrity despite doubling the number of vehicles from 4 to 8. On the other hand, MQTT, suffered quite a bit, with 29 out of 100 test runs seeing data integrity loss.

That is not to say that MQTT is unsuited for the task however. In our experiment, we subjected both protocols to the full VECFrame pipeline, which forces a wait for complete data retrieval before moving forward with the data to the processing containers. This was to ensure that we have no faulty data as using such can result in serious consequences.

By taking the strength of both Wget and MQTT, we implement both to work in tandem to reach a optimized balance for the framework. Wget facilitates the data transfer to ensure data integrity, while MQTT disseminates the results back to the vehicles with small latency as seen in Table I.

VII. Conclusions

Autonomous vehicles will be connected with edge computing infrastructures which provide computing and storage resources for AI-driven CAV services and applications. In this paper, we present VECFrame, a CAV-edge framework which provides efficient multi-vehicle and/or vehicle-traffic infrastructure with cooperative edge computing services. VECFrame offers a series of scalable and reliable methods for vehicle data aggregation, sensor data fusion, cooperative object detection, and results dissemination. We use CAV cooperative perception as a use case to present and evaluate the communication subsystem and the edge computing workflow in VECFrame. Experimental results on a real CAV test platform show the promising results of VECFrame in exploiting edge resources for extending vehicles' perception range and enhancing object detection accuracy for both self driving and traffic management with minimal latency.

In VECFrame, we assume that the industry sector will provide edge as a service, which is more general and effective

than the way that current vehicles upload their data from vehicles[1]. We see security, optimized scheduling and energy computation trade-off as opportunities for our future research.

Edge computing and CAVs are both rapidly developing fields of interest. Our framework benefits both fields in that VECFrame provides a holistic framework with edge-based mechanisms to enable vehicle-vehicle and vehicle-infrastructure cooperative services. With more autonomous vehicles running on road and more data shared among vehicles, scalable cooperative computation and real-time decision making can be achieved by using VECFrame.

ACKNOWLEDGMENT

This work has been supported in part by the National Science Foundation grants CNS-2113805, CNS-1852134, OAC-2017564, ECCS-2010332, CNS-2037982, CNS-1563750, and a research grant from Fujitsu.

REFERENCES

[1] Autopilot – tesla. https://www.tesla.com/autopilot.

[2] For self-driving cars, there's big meaning behind one big number: 4 terabytes. https://newsroom.intel.com/editorials/self-driving-cars-big-meaning-behind-one-number-4-terabytes/.

[3] Nvidia drive agx developer kit | nvidia developer. https://developer.nvidia.com/drive/drive-agx.

[4] wget. https://www.gnu.org/software/wget//.

[5] Sae levels of driving automation™ refined for clarity and international audience. https://www.sae.org/blog/sae-j3016-update, 2021. (Accessed on 11/12/2021).

[6] E. Arnold, M. Dianati, and R. de Temple. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors, 2019.

[7] J. Baber, J. Kolodko, T. Noel, M. Parent, and L. Vlacic. Cooperative autonomous driving: intelligent vehicles sharing city roads. *IEEE Robotics & Automation Magazine*, 12(1):44–49, 2005.

[8] D. Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[9] M. Chen and Y. Hao. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, 36(3):587–597, 2018.

[10] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu. F-cooper: feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *ACM/IEEE Symposium on Edge Computing (SEC)*, 2019.

[11] Q. Chen, S. Tang, Q. Yang, and S. Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019.

[12] J. Du, L. Zhao, J. Feng, and X. Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Trans. on Communications*, 66(4):1594–1608, 2018.

[13] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, F. Puetzold, and C. Wohler. Autonomous driving goes downtown. *IEEE Intelligent Systems and Their Applications*, 13(6):40–48, 1998.

[14] Fred Lambert. Understanding the fatal tesla accident on autopilot and the nhtsa probe. https://electrek.co/2016/07/01/understanding-fatal-tesla-accident-autopilot-nhtsa-probe/, 2016.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] A. Gupta, R. Christie, and P. Manjula. Scalability in internet of things: features, techniques and research challenges. *Int. J. Comput. Intell. Res*, 13(7):1617–1627, 2017.

[17] Y. He, N. Zhao, and H. Yin. Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 67(1):44–55, 2017.

[18] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *International Conference on Communication Systems Software and Middleware*, 2008.

[19] R. Hussain and S. Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2018.

[20] J. Kaiser and M. Mock. Implementing the real-time publisher/subscriber model on the controller area network (can). In *IEEE Intl. Symposium on Real-Time Distributed Computing (ISORC)*, 1999.

[21] S. Kato, S. Tsugawa, K. Tokuda, T. Matsui, and H. Fujii. Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications. *IEEE Trans. on Intelligent Transportation Systems*, 3(3):155–161, 2002.

[22] J. B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[23] H. Li, K. Ota, and M. Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *Network*, 32(1):96–101, 2018.

[24] D. Liu, B. Chen, C. Yang, and A. F. Molisch. Caching at the wireless edge: design aspects, challenges, and future directions. *IEEE Communications Magazine*, 54(9):22–28, 2016.

[25] 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*, 107(8):1697–1716, 2019.

[26] Y. Liu, J. Zhang, Y. Li, P. Hansen, and J. Wang. Human–computer collaborative interaction design of intelligent vehicle—a case study of hmi of adaptive cruise control. 2021.

[27] Q. Luo, C. Li, T. H. Luan, and W. Shi. Collaborative data scheduling for vehicular edge computing via deep reinforcement learning. *IEEE Internet of Things Journal*, 2020.

[28] Q. Luo, C. Li, T. H. Luan, and W. Shi. Edgevcd: Intelligent algorithm inspired content distribution in vehicular edge computing network. *IEEE Internet of Things Journal*, 2020.

[29] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov. Real-time computer vision with opencv. *Communications of the ACM*, 55(6):61–69, 2012.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[31] B. Schwarz. Lidar: Mapping the world in 3d. *Nature Photonics*, 4(7):429, 2010.

[32] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[33] Y. Shi, Q. Han, W. Shen, and X. Wang. A multi-layer collaboration framework for industrial parks with 5g vehicle-to-everything networks. *Engineering*, 2021.

[34] J. Tang, S. Liu, L. Liu, B. Yu, and W. Shi. Lopecs: A low-power edge computing system for real-time autonomous driving services. *IEEE Access*, 8:30467–30479, 2020.

[35] Wired. Why tesla's autopilot can't see a stopped firetruck. https://www.wired.com/story/tesla-autopilot-why-crash-radar/, 2018.

[36] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan. Fcss: Fog computing based content-aware filtering for security services in information centric social networks. *IEEE Transactions on Emerging Topics in computing*, 2017.

[37] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, and X. Shen. Internet of vehicles in big data era. *IEEE/CAA Journal of Automatica Sinica*, 5(1):19–35, Jan 2018.

[38] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. Lavea: Latency-aware video analytics on edge computing platform. In *ACM/IEEE Symposium on Edge Computing (SEC)*, 2017.

[39] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu. Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet of Things Journal*, 5(4):2633–2645, 2017.

[40] Q. Zhang, Y. Wang, X. Zhang, L. Liu, X. Wu, W. Shi, and H. Zhong. Openvdap: An open vehicular data analytics platform for cavs. In *IEEE Intl. Conference on Distributed Computing Systems (ICDCS)*, 2018.

[41] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong. Distributed collaborative execution on the edges and its application to amber alerts. *IEEE Internet of Things Journal*, 5(5):3580–3593, 2018.

[42] L. Zhao, X. Li, B. Gu, Z. Zhou, S. Mumtaz, V. Frascolla, H. Gacanin, M. I. Ashraf, J. Rodriguez, M. Yang, et al. Vehicular communications: Standardization and open issues. *IEEE Communications Standards Magazine*, 2(4):74–80, 2018.

[43] C. Zhu, G. Pastor, Y. Xiao, and A. Ylajaaski. Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges. *IEEE Communications Magazine*, 56(10):58–63, 2018.