2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD) | 978-1-6654-4594-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/METROCAD51599.2021.00011

Vehicular Edge Computing for Multi-Vehicle Perception

Sihai Tang
Computer Sci. & Engg. Dept.
University of North Texas
Denton, Texas 76203
SihaiTang@my.unt.edu

Zhaochen Gu
Computer Sci. & Engg. Dept.
University of North Texas
Denton, Texas 76203
ZhaochenGu@my.unt.edu

Song Fu
Computer Sci. & Engg. Dept.
University of North Texas
Denton, Texas 76203
Song.Fu@unt.edu

Qing Yang
Computer Sci. & Engg. Dept.
University of North Texas
Denton, Texas 76203
Qing.Yang@unt.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 Scheduling Pipeline for connected and autonomous vehicles (CAVs) exploring scheduling optimization, pipeline design and vehicle to edge interactions. Through our pipeline, 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, our pipeline facilitates data transfer and fusion for cooperative object detection of multiple vehicles. Through real-world experiments, we evaluate the performance and robustness of our pipeline on different device architectures and under different scenarios. We demonstrate that our pipeline achieves a real-time deadline capable edge to vehicle interaction via vehicle-edge data transfer and on-edge computation.**

***Index Terms*—Resource Profiling, Connected and Autonomous Vehicles, Workload optimization**

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], [32], [38], with amber alert detection and collision avoidance being the most prominent [39].

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 [6], [9], [10], 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.

With these limitations, autonomous vehicle manufacturers opt to use cellular connectivity to facilitate data between vehicles and computing platforms [1]. With 5G infrastructure yet to reach maturity in many regions of the country, the information type, size and variety are often restricted due to the availability and location. We typically see the use of dedicated software and hardware such as dedicated short range communication (DSRC) as a channel of data dissemination between different vehicles, but this too is limited to the information type and size. While DSRC can broadcast the GPS, IMU and speed information for other vehicles to stay aware, large sensor data from HD cameras and LiDAR requires higher bandwidth which DSRC cannot provide. Those 2D images and 3D point clouds are useful for improving the perception range and accuracy of vehicles leveraging edge computing [9].

Extensive research on how to improve the safety for all parties involved in a autonomous car points to CAV as the solution [33], [34]. Stemming from this, with communication between vehicles, we open up the possibility of cooperative perception, which eliminates many faults of a single vehicle operating on its own sensors. However, as is with all things, we face challenges in the area of cooperative perception as well. Works such as [5], [9], [10], [17] present the limitations as well as the advantages of sharing sensor data between vehicles.

While these works greatly improve the outlook of autonomous vehicles, they still rely on the use of traditional platforms of data sharing such as cellular or DSRC. With the emerging edge computing paradigm, we find that although works such as [9] are suitable for edge, there is no end-to-end edge system that can address the challenges of using the edge as well as the ability to fully facilitate the process of cooperative perception on the edge.

In this paper, we design and evaluate our scheduling pipeline for the transfer and fusion of different types of data between vehicles and edge nodes to achieve cooperative perception. We propose the use of edge nodes as a targeted and purposed system to facilitate the exchange of large sensor data towards safer driving. Through our pipeline, the data, generated by on-board

978-1-6654-4594-8/21/$31.00 ©2021 IEEE
DOI 10.1109/MetroCAD51599.2021.00011

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, our pipeline facilitates data transfer and fusion for cooperative object detection of multiple vehicles. Through real-world experiments, we evaluate the performance and robustness of our pipeline on different device architectures and under different scenarios. We demonstrate that our pipeline achieves a real-time deadline capable edge to vehicle interaction via vehicle-edge data transfer and on-edge computation.

The rest of the paper is organized as follows. Section II discusses the related research. Section III details vehicular edge computing and our edge scheduling pipeline design. Section IV presents the performance results from edge computing for multi-vehicle cooperative perception. Section V and Section VI discuss our design and results and point out directions of future research. Section VII concludes the paper.

II. Related Work

In this section, we investigate the existing literature on vehicular edge computing for scheduling and pipeline design in the connected autonomous vehicle (CAV).

Edge for vehicles. [25] provided us a comprehensive review of development and challenges of autonomous vehicle in edge environment. [16] proposed a vehicle-to-vehicle communication enhancement scheme by introduced the hierarchical edge-based preemptive route creation, two-stage learning and context-aware edge selection approach to improve the packet forwarding performance. As edge computing becomes a mainstream solution to process data remotely for autonomous, [28] investigated the service migration and resource management from intra- and inter-tier communication in edge and fog computing. In addition, many existing works mentioned about edge for vehicles but focused on MEC (Mobile-Edge-Computing). [14] an [18] solved the offloading and caching computing in vehicle networks for mobile-based edge separately.

Resource allocation and offloading. Resource allocation plays an important role in CAV system optimization problem. [27] studied the tradeoff between execution time and energy consumption for the problem of computation offloading. They also designed a game theory model to minimize the combination of energy overhead and delay. Similarly, Zhang [37] designed a system to reduce the weighted cost of time latency and energy consumption using mobile devices as the edge ends. [22] also proposed a computation offloading technique in terms of energy-efficiency in edge cloud computing. [13] proposed a distributed offloading scheme by helping multiple users learn their long-term offloading strategies and proved a Nash equilibrium can be achieved. [21] propose a method to allocate resources in a uncertain conditions improve the reliability of Vehicle-to-infrastructure system.

Scheduling. Scheduling on edge side and between edge to vehicle is a prevalent topic in recent years. Most of the paper focused on communication and scheduling strategy on mobile-based edge ends. [26] proposed a framework to formulate the communication and resource computation jointly and promote an optimal strategy on data scheduling using deep reinforcement learning approach. [11] optimized the task scheduling by designed a dynamic scheduling scheme in regard to queue-based and time-based approach in order to allocate tasks in hybrid mobile environments. [23] designed a time-driven workflow scheduling mechanism to efficiently improve the completion time of reasoning tasks in edge environments by applying the Markov decision process and Q-learning algorithm in their simulated annealing. In addition, [36] also applied Markov decision process in their method but combine with other optimization algorithm in deep reinforcement learning. He also implemented a representative features extraction by merging parameter-shared network architecture together with convolutional neural network.

III. Edge Computing for Connected Autonomous Vehicles

A main advantage of employing edge computing compared with cloud computing to CAVs is the lower latency [30]. Although the cloud has more resources and is more powerful than an edge device, processing data on edge devices is more responsive than forwarding data over the Internet to the cloud. For example, GM(general motors) has recently announced their new electric vehicles line up with improved ultium batteries that is supposed to eliminate emissions as well as be more road friendly [3]. This move towards better on vehicle energy allows for more energy to be devoted towards extra protocols, such as edge communication. However, there is still the need for an efficient approach to allow for mass adoption of such protocols, especially due to the persistent nature of such protocols.

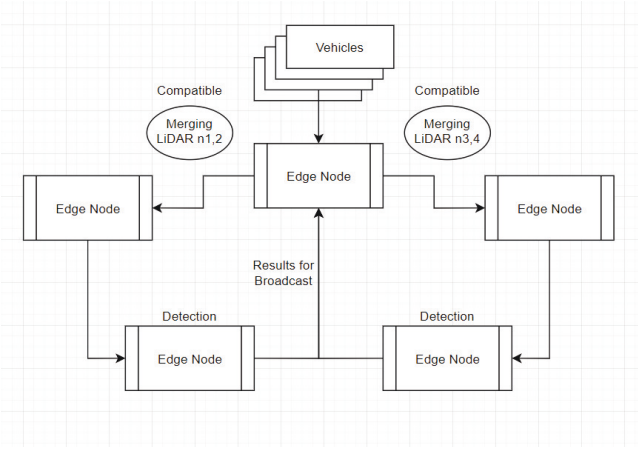

Fig. 1. The Vehicles designate cars with the capability of requesting service from the edge node network. As the requests are received, the edge node handling the message transactions will cluster the requests into potential groups for further processing. Through this pipeline, a wide array of models can then be deployed to the correct vehicle cluster for faster and more accurate processing.

We argue that edge computing is probably the most suitable computational paradigm to facilitate heterogeneous, large-

scale, and agile data processing for CAVs [24]. Towards this end, we design a data processing pipeline that utilizes the benefits of vehicles' close proximity to edge nodes to achieve low-latency data processing and real-time decision making, thereby opening up possibilities for various CAV applications, including cooperative perception where sensor data from multiple vehicles are exchanged and fused to detect objects in a wider range and with a better accuracy [9], [10], [12], [17], [20].

A. Data Communication in Vehicular Edge Computing for CAVs

Before designing an edge computing pipeline for CAVs, it is important to understand the state of the art of CAV technologies, as well as the barriers that hamper the deployment of edge computing. Here, we briefly review the current data exchange technologies used by CAVs, including their major advantages and limitations. Starting off, we look at the more traditional methods such as DSRC. This allows for quick inter-vehicular communication of small packets with other similarly equipped vehicles. However, the downsides are also clear in that DSRC cannot support high amounts of data flow. Recently, more efforts are focused on bringing vehicle-to-everything (V2X) communication for autonomous driving closer to reality. With companies such as Verizon and T-Mobile experimenting with 5G in certain regions, it is no longer a dream to have a connected environment for connected and autonomous driving [4]. Although, despite the higher bandwidth of the new 5G network, the transfer of large amounts of HD map data and other data between multiple vehicles will stress the local infrastructure unnecessarily. To support the maintenance of data quality, it is estimated that around 1 terabyte of data will be need to collected and sent over the period of one hour [31]. The transfer of HD maps and other crucial sensor data between vehicles can be taxing, but if it is instead routed through a trusted edge device with supporting applications, then we would avoid the issues presented.

However, through utilizing the edge device, we open up more challenges. As previously mentioned, trust, different data types, hardware and software compatibility and versioning support are all major impediments to an efficient vehicle to edge data exchange paradigm. Therefore, it is crucial to design efficient data transmission and data processing pipelines to enable edge computing for CAVs.

For other vehicular communication solutions, e.g., DSRC [19] and WiFi Direct [7], we face more stringent restrictions as both bandwidth as well as real-time deadlines are heavily stressed without a proper processing pipeline on the edge.

From vehicles to the cloud, there exists routing latency, and in this situation, a direct connection between vehicles and edge servers provides less communication latency [25], [32]. Various other barriers such as cooperative perception and object detection arise when vehicles are reliant on the cloud to provide computational power towards such purposes. Not only does the cloud need to calculate the data, it will also need to send out requests for the vehicles in the proximity of the requesting vehicle. This multi-stage approach drastically increases the overhead of performing such tasks.

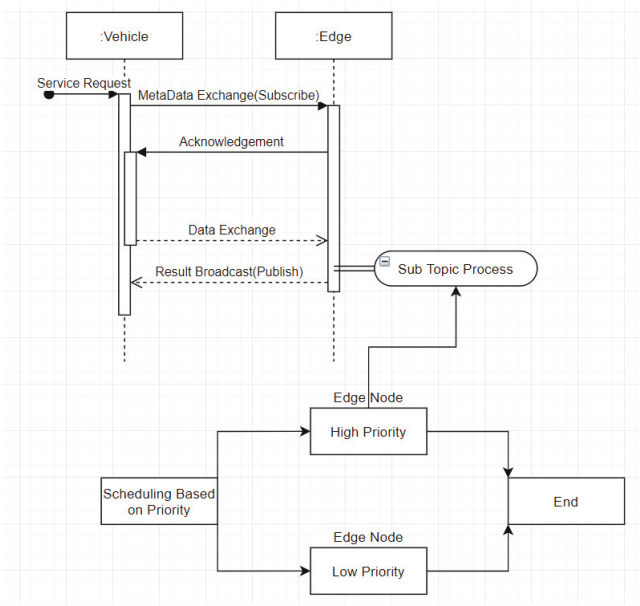

Fig. 2. This pipeline details the exchange between the vehicle and the edge node in detail. The vehicle is assumed to have service access to the specified edge node as a part of a service agreement between providers. The vehicle can send the request contains the necessary metadata to the edge in the handshaking process. After receiving acknowledgement from the edge, the data transfer can then begin. After which, the edge will then designate the vehicle request based on cluster and priority through a scheduling optimization process before the work can begin. The final results are then broadcasted back to the vehicle in need.

B. Design Issues with Vehicular Edge Computing

As mentioned before, with the plethora of new cooperative perception techniques designed for CAVs [6], [9], [10], we require far more data to be transmitted than what the traditional data transmission can support. For example, raw 3D LiDAR data [10] and feature maps generated from the CNN models [9] on multiple vehicles need to be consolidated through the fusion process, however, it will require each vehicle handle the handshaking process individually with every new vehicle. The reason why handshaking is such an issue revolves around the task at hand as well as the vehicles that are communicating. Take for example a top end autonomous vehicle and a lower end consumer model with the basic sensors. Should these two vehicles request each other with data, the lower end will not have the resources to process the data from the higher end model. This handshake will result in a failure due to hardware specifications. Another barrier exists in the number of vehicles trying to handshake. With no guarantee of a unified standard, the most challenging issue at hand is trust. Many manufactures do not trust the information from another vehicle not made by them due to liability issues, so they will not succeed in a handshake for data, limiting the scope of available sources.

Aside from these barriers, the overhead of repeatedly trying to establish connection with surrounding vehicles will increase with the number of vehicles. Just as a Distributed Denial of Service attack can slow or halt access to a service all together, the same situation is possible with repeated handshaking due to connection or trust issues. This is a very costly process both in on-board unit (OBU) computational resources as well as network bandwidth, the OBU refers to the computing unit on board the vehicle handling inference, vehicle status, and other crucial tasks for autonomous driving.

This problem is compounded by the fact that significant amount of data, at least tens of megabits per second per vehicle, need to be transferred and processed within a CAV system [10]. If stationary sensor data, generated from roadside infrastructures, is included in the fusion process, more localized data processing is required [6].

Therefore, it is crucial for an edge processing pipeline to be data friendly in order to facilitate these data-heavy communication and computation tasks.

With traditional methods, DSRC will keep broadcasting WAVE short message (WSM) at a set frequency [19], regardless of whether the information broadcasted is acknowledged by other cars; this holds both benefits and detriments. Although it simplifies the data transmission protocol, which is beneficial for safety message transmission, it does not support the transmission of raw (or extracted) sensor data. While both hold irrefutable merits, we believe that sharing either raw or extracted data among autonomous vehicles is of great value, as both can be used towards better perception and driving safety. An encompassing system that addresses the aforementioned new challenges is still missing in the literature.

C. Task Scheduling in Vehicular Edge Computing

To approach issue, the aforementioned encompassing system needs both efficiency and stability to be a reliable platform to serve critical operations such as real-time driving. Alongside the challenges and current techniques, works that detail new uses of edge for CAVs such as [8], [35], open up new challenges; the first work examine the security aspect where as the second work studies the effects of CAVs on a platoon of vehicles for collaboration and fueling purposes.

Due to the massive scope of effective edge usage when it comes to CAVs, we design an optimization algorithm centered around the Edge side scheduling, with the primary focus being on the profiling and establishing the baseline performance of the native pipeline from F-Cooper. Through this, we make the first step towards eliminating the challenges of safe collaborative driving through the use of Edge, with the focus on pipeline design and scheduling.

The problem formulation is as follows. At any given time slot, the data of each vehicle can be:

- Processed locally: select features to send to edge (f_i)
- In queue to be uploaded to the Edge: time restricted based on speed (Certain time frame before the location data is irrelevant.) (d_i)
- On the Edge awaiting processing. (s_i)
- Processed by the Edge awaiting either deletion or Archiving. (q_i)

At any given time, the Edge node will consider the following:

- Resources available (Possible consideration of Down sampling) (R_j)
- Deadline assignment and tracking for each vehicle (D_j)
- Tasks awaiting execution (Estimated execution time) (T_j)
- Tasks in execution (Estimated completion time) (E_j)

Factoring all the possible elements for analysis, we use the following general equation:

$$\begin{aligned} \underset{X}{\text{Optimize}} \quad & f_0(X_{i,j}),\ (i,j) \in \omega, \\ \text{subject to} \quad & X_{ij} = M_{ij}, \\ & X \succeq 0, \\ \text{Where} \quad & M_{ij} = (d_i, s_i, f_i, q_i) \uplus (R_j, D_j, T_j, E_j), \end{aligned} \tag{1}$$

where X is the set of all incoming requests with i representing the vehicle and j representing the edge node. As long as both vehicle and edge node are within service range , then for each element of X, the corresponding set $M_{i,j}$ will be processed as long as it is not empty. Each set of $M_{i,j}$ will be consisted of elements from both the vehicle and edge.

IV. Vehicular Edge Computing for Cooperative Perception

We conduct experiments to evaluate the run-time process of F-Cooper using the open-source code cited in [9]. Through this, the vehicles are able to request for an extension of F-Cooper to be performed. We then profile F-Cooper through analysis of the current pipeline implemented to identify potential points of optimization.

As F-Cooper is inherently a lightweight process capable of running on an edge device, it has several components that we analyse through our experiment. Since F-Cooper supports the use of the GPU, we prioritize the utilization of the GPU over the CPU in our experiment. In our experiment, we will profile the following elements of the process in order to establish a good baseline for future work on the optimization and scheduling:

- GPU usage
- GPU energy consumption
- Total Run-time for a complete processing run of one frame of data
- Algorithmic break down of the components as well as their formulations

The overall condensed pipeline of the F-Cooper is shown in Fig 3. As shown in the figure, we can see that the overall processes of the F-Cooper is fairly linear in nature. In our designed pipeline architecture, the data preprocessing for F-Cooper is not necessary as the handshaking assumes that the data is pre-compatible. Also, the model training step will also be excluded as the Edge is assumed to have access to the latest models available for servicing the compatible vehicles. With these two big limiting factors removed from the F-Cooper process, we now look specifically at the following steps of

Nvidia GPU Status (Before)					
GPU Information	Name	Temperature	Performance State	Persistence Mode	Fan Speed
	GeForce GTX 1060	39C	P2	Off	38%
Usage Information	Power Usage	Power Capacity	Memory Usage	Memory Capacity	Volatile GPU Utilization—
	27W	120W	301 MiB	6075 MiB	1%
Processes Information	GPU Index	PID	Type	Process Name	GPU Memory Usage
	0	1314	G	/usr/lib/xorg/Xorg	22 MiB
	0	2000	G	/usr/lib/xorg/Xorg	190 MiB
	0	5238	C	Python3	63 MiB

TABLE I

THE STATUS OF THE GPU WHEN NOT UNDER A WORKLOAD. HERE THE IDLE TEMPERATURE OF HAVING F-COOPER LOADED IN THE BACKGROUND IS AROUND 39 CELSIUS WITH A POWER DRAW OF AROUND 27 WATTS. THE TOTAL MEMORY USAGE IS ALSO VERY LOW AT ONLY 300 MB.

Nvidia GPU Status (After)					
GPU Information	Name	Temperature	Performance State	Persistence Mode	Fan Speed
	GeForce GTX 1060	40C	P2	Off	38%
Usage Information	Power Usage	Power Capacity	Memory Usage	Memory Capacity	Volatile GPU Utilization—
	27W	120W	1927 MiB	6075 MiB	32%
Processes Information	GPU Index	PID	Type	Process Name	GPU Memory Usage
	0	1314	G	/usr/lib/xorg/Xorg	22 MiB
	0	2000	G	/usr/lib/xorg/Xorg	190 MiB
	0	5238	C	Python3	1689 MiB

TABLE II

THE STATUS OF THE GPU WHEN UNDER A SINGLE F-COOPER WORKLOAD. HERE THE TEMPERATURE IS UP BY 1 DEGREES CELSIUS OVER THE IDLE TEMPERATURE AND THE TOTAL MEMORY USAGE IS UP TO AROUND 2 GB AS COMPARED TO THE 300MB OF WHEN IT WAS IDLE.

voxel generation and forwarding as well as the inference speed. However, as the main portion of training the model for accuracy is calculation of the loss function, to gouge the difference between the prediction to the ground true, we briefly examine the process to estimate the impact of running such on an Edge device. The loss function of F-Cooper used for evaluation purposes is defined as follows. Suppose the model proposes N_{pos} positive anchors and N_{neg} negative anchors, they define the loss function as follows:

$$\begin{aligned} L = \alpha \frac{1}{N_{neg}} \sum_{i=1}^{N_{neg}} L_{cls}\left(p_{neg}^{i}, 0\right) \\ + \beta \frac{1}{N_{pos}} \sum_{i=1}^{N_{pos}} L_{cls}\left(p_{pos}^{i}, 1\right) \\ + \frac{1}{N_{pos}} \sum_{i=1}^{N_{neg}} L_{reg}\left(P^{i}, G^{i}\right) \end{aligned} \tag{2}$$

where p_{neg}^{i} and p_{pos}^{i} are the probability of positive anchors and negative anchors respectively, and N_{neg} and N_{pos} denote the number of proposed negative and positive anchors respectively. In regression loss, G^{i} indicates the ith ground truth while P^{i} means the corresponding predicted anchor. They use α and β to balance these three losses. They also employ a binary cross entropy loss for classification Loss and Smooth-L1 loss function [15], [29].

As we can see from the equation, the amount of processing time will increase linearly based on the amount of anchors proposed through the Region Proposal process. This relationship between the input data and the amount of processing time required to run such calculations indicate that as more objects enter the area of service, the more processing power is needed should we require model updates during downtime.

A. GPU Usage

F-Cooper considers both the CPU and the GPU as a potential source of computational power, however, to focus on more realistic profiling, we opted to use the Nvidia GeForce GTX 1060 GPU. This GPU has a maximum power draw of 120 watts and a total of 6 gigabytes of memory.

In our first task, we loaded all the process threads into the edge system and profiled the overall GPU status. As we see in Table. I, the status of the GPU when not under a workload. Here the Idle temperature of having F-Cooper loaded in the background is around 39 Celsius with a power draw of around 27 watts. The total memory usage is also very low at only 300 MB.

With this data as our baseline, we move on to the full profiling of the GPU with F-Cooper running one frame of data inference.

The status of the GPU when under a single F-Cooper workload is shown in Table. II. Here the temperature is up by 1 degrees Celsius over the idle temperature and the total memory usage is up to around 2 GB as compared to the 300MB of when it was idle. This increase in both temperature and memory usage is a big increase when taking into consideration that we are just running inference for one frame of data.

As a continuous stream of data is fed through the F-Cooper's native pipeline, the amount of processes being loaded comes to around 1.7 GB, which is very large to run a inference process.

In addition to the memory increase, we also saw the increase in temperature. As one frame is very small, a realistic workload will ramp up the temperature and thus force the GPU to draw more power to both cool and process the workload more efficiently. We this that this is a key factor to optimizing the performance of the pipeline.

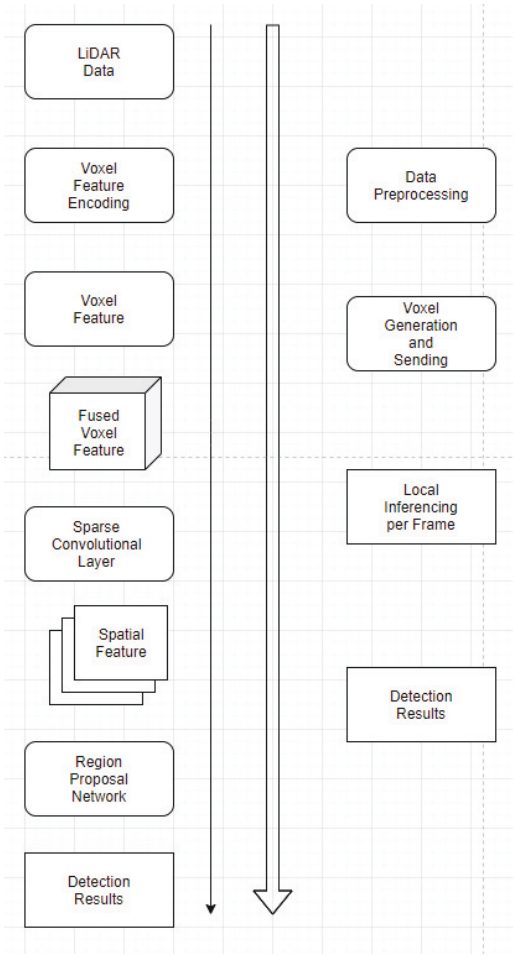

Fig. 3. Architecture of the feature based cooperative perception (F-Cooper). F-Cooper has multiple vehicles' (using two here for illustration) LiDAR data inputs which are processed by the VFE layers respectively to generate voxel features. To fuse 3D features from two cars, two fusion paradigms are designed: voxel features fusion and spatial features fusion. In Paradigm I, two sets of voxel features are fused first and then spatial feature maps are generated. In Paradigm II, spatial features are first obtained locally on individual vehicles and then fused together to generate the ultimate feature maps. Symbol $\bigoplus$ indicates where the fusion takes place in each paradigm. An RPN is employed for object detection on the ultimate feature maps in both paradigms. We use dashed arrows to denote data flow and bold red lines to present fusion connections. Best viewed in color.

B. Run Time Average

Just the physical hardware usage is not enough for a good pipeline design, so we also conduct a profiling for the overall speed of F-Cooper broken down into the following parts:

- Label Generation
- Total Run-time for a complete processing run of one frame of data
- Algorithmic break down of the components as well as their formulations

As seen in Table. III, the average time for a single F-Cooper run is broken down. With Label generation and evaluation as the two main components of interest. We observe the rapid generation of the labels at around 0.85 iterations of tasks per second. Following the label generation, the evaluation step comes next. First, the Voxels are forwarded from the neural network at a rate of 0.048 seconds per example. This is the time that it takes for the native pipeline to extract and initiate the process of sending the voxels to the target. After the extraction and forwarding of the voxels per example, the native pipeline then continues with the local inference portion, to generate a prediction for the example at hand. As we see in the figure, the average post process inference time per example is around 0.191 seconds, which is fairly long when compared to the voxel extraction and forwarding step.

Remain Numbers of Infos	Avg Forward Time (per example)	Avg Post-process Time (per example)
1	0.048 seconds	0.191 seconds

TABLE III
THE AVERAGE TIME FOR A SINGLE F-COOPER RUN IS BROKEN DOWN. WITH LABEL GENERATION AND EVALUATION AS THE TWO MAIN COMPONENTS OF INTEREST.

However, as we only focus on profiling the native pipeline of F-Cooper, the relative time it takes to receive and also inference the example timing is not tested.

V. DISCUSSIONS

The profiling of the F-Cooper native pipeline has shown possibilities for the use of an algorithm based scheduling optimization. For example, the workload of the native pipe gives credence to the heavy weight nature of the entire process.

In our approach, the background resource usage can be optimized for much less usage towards the task at hand. For example, in our profiling, the native F-Cooper pipe shows around 30 percent GPU usage for just the inference of a single frame. This is most likely using the GPU for tasks that can be routed to the CPU, such as pre-loading the model weights and other similar tasks.

Further observations made based on the profiling results indicate that while F-Cooper achieves real-time speeds, it is still linear in nature for the job execution order. As the voxels forwarding is not a crucial step that is depended on by the evaluation and prediction tasks later on, it can be separated for variable scheduling instead of a static sequential order of execution.

VI. POTENTIAL RESEARCH

Our results from the experiment proves that there exists room for optimization in the native pipeline for F-Cooper. Our proposed pipeline takes much of the heavy workload that F-Cooper requires and boils it down to a constant background process rather than the current design of load on use.

Additional factors that play a role in slowing down F-Cooper can be explored as possible optimization tasks to further extend the possibility of opening research potentials for the integration between edge and autonomous vehicles.

VII. Conclusions

In this paper, we identify that the existing literature does not focus on the approach of integrating edge with autonomous vehicles specifically. While efforts have been made towards this end, we do not see the scalability for the existing works. We believe that through optimization, a pipeline can be designed and formalized for current and future efforts, thus allowing for the easy transition for works such as F-Cooper to fully migrate to various edge platforms without loss of performance.

In our experiment, we profile F-Cooper. Based on our experimental analysis, we find that there exists many opportunities to apply our proposed algorithm to optimize the existing native pipeline of F-Cooper. We formulate a formal method that allows for such a process to be adopted in this field.

Acknowledgment

This work has been supported in part by the National Science Foundation grants CNS-1852134, OAC-2017564, ECCS-2010332, CNS-2037982, and CNS-1563750. We thank the anonymous reviewers for their constructive comments, which helped us improve this paper.

References

[1] Connectivity — tesla. https://www.tesla.com/support/connectivity.

[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] Gm's path to an all-electric future — general motors. https://www.gm.com/electric-vehicles.html.

[4] Verizon vs at&t vs t-mobile vs sprint: Choose the best 5g carrier - cnet. https://www.cnet.com/how-to/verizon-vs-at-t-vs-t-mobile-vs-sprint-choose-the-best-5g-carrier/.

[5] O. Altintas and T. Higuchi. Multi-level hybrid vehicle-to-anything communications for cooperative perception, Oct. 24 2019. US Patent App. 15/958,969.

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

[7] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. Device-to-device communications with wi-fi direct: overview and experimentation. *IEEE wireless communications*, 20(3):96–104, 2013.

[8] C. Chen, J. Jiang, N. Lv, and S. Li. An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge. *IEEE Access*, 8:99059–99069, 2020.

[9] 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.

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

[11] X. Chen, N. Thomas, T. Zhan, and J. Ding. A hybrid task scheduling scheme for heterogeneous vehicular edge systems. *IEEE Access*, 7:117088–117099, 2019.

[12] L. Ding, Y. Wang, P. Wu, L. Li, and J. Zhang. Kinematic information aided user-centric 5g vehicular networks in support of cooperative perception for automated driving. *IEEE Access*, 7:40195–40209, 2019.

[13] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin. Learning for computation offloading in mobile edge computing. *IEEE Transactions on Communications*, 66(12):6353–6367, 2018.

[14] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu. Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. *IEEE Transactions on Vehicular Technology*, 68(2):1079–1092, 2018.

[15] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[16] S. Guleng, C. Wu, Z. Liu, and X. Chen. Edge-based v2x communications with big data intelligence. *IEEE Access*, 8:8603–8613, 2020.

[17] T. Higuchi, M. Giordani, A. Zanella, M. Zorzi, and O. Altintas. Value-anticipating v2v communications for cooperative perception. In *IEEE Intelligent Vehicles Symposium*, 2019.

[18] R. Q. Hu et al. Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 67(11):10190–10203, 2018.

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

[20] S. Kim and W. Liu. Cooperative autonomous driving: A mirror neuron inspired intention awareness and cooperative perception approach. *IEEE Intelligent Transportation Systems Magazine*, 8(3):23–32, 2016.

[21] A. Kovalenko, R. F. Hussain, O. Semiari, and M. A. Salehi. Robust resource allocation using edge computing for vehicle to infrastructure (v2i) networks. In *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–6. IEEE, 2019.

[22] X. Li, Y. Dang, M. Aazam, X. Peng, T. Chen, and C. Chen. Energy-efficient computation offloading in vehicular edge cloud computing. *IEEE Access*, 8:37632–37644, 2020.

[23] K. Lin, B. Lin, X. Chen, Y. Lu, Z. Huang, and Y. Mo. A time-driven workflow scheduling strategy for reasoning tasks of autonomous driving in edge environment. In *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 124–131. IEEE, 2019.

[24] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi. Computing systems for autonomous driving: State-of-the-art and challenges. *IEEE Internet of Things Journal*, pages 1–1, 2020.

[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] 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.

[27] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci. Computation offloading game for an uav network in mobile edge computing. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[28] L. Pacheco, H. Oliveira, D. Rosário, E. Cerqueira, L. Villas, and T. Braun. Service migration for connected autonomous vehicles. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[30] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[31] H. G. Seif and X. Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2):159–162, 2016.

[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] A. Talebian and S. Mishra. Predicting the adoption of connected autonomous vehicles: A new approach based on the theory of diffusion of innovations. *Transportation Research Part C: Emerging Technologies*, 95:363–380, 2018.

[34] C. Wang, S. Gong, A. Zhou, T. Li, and S. Peeta. Cooperative adaptive cruise control for connected autonomous vehicles by factoring communication-related constraints. *Transportation Research Part C: Emerging Technologies*, 2019.

[35] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang. Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles. *IEEE Wireless Communications*, 27(3):24–30, 2020.

[36] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu. Deep reinforcement learning-based offloading scheduling for vehicular edge computing. *IEEE Internet of Things Journal*, 2020.

[37] K. Zhang, X. Gui, D. Ren, and D. Li. Energy-latency tradeoff for computation offloading in uav-assisted multi-access edge computing system. *IEEE Internet of Things Journal*, 2020.

[38] 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 *Distributed Computing Systems (ICDCS), 2017 IEEE 38th International Conference on. IEEE*, 2018.

[39] 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.