Lightweight Edge Intelligence Empowered Near-crash Detection Towards Real-time Vehicle Event Logging

Ruimin Ke *Member, IEEE*, Zhiyong Cui, Yanlong Chen, Meixin Zhu,
Hao Yang *Graduate Student Member, IEEE*, Yifan Zhuang, Yinhai Wang *Fellow, IEEE*

***Abstract*—A major role of automated vehicles is that vehicles serve as mobile sensors for event detection and data collection, which support tactical automation in autonomous driving and post-analysis for traffic safety. However, most data collected during regular operations of vehicles are not of interest, while it costs a large amount of computation, communication, and storage resources on the cloud servers. Vehicular edge computing has emerged as a promising paradigm to balance these high costs in traditional cloud computing. But edge computers often have limited resources to support the high efficiency and intelligence of advanced vehicular functions. Motivated by the existing challenges and new concepts, this paper proposes and tests a lightweight edge intelligence framework for vehicle event detection and logging that runs in an event-based and real-time manner. Specifically, this paper takes vehicle-vehicle and vehicle-pedestrian near-crashes as the events of interest. The lightweight algorithm design of modeling the bounding boxes in object detection/tracking enables real-time edge intelligence onboard a vehicle; The event-based data logging mechanism eliminates redundant data onboard and integrates multi-source information for individual near-crash events. Comprehensive open-road tests on four transit vehicles have been conducted.**

***Index Terms*—Autonomous driving, edge artificial intelligence, vehicle near-crash, object and event detection and response, real-time system.**

I. Introduction

Object and event detection and response (OEDR) is a subtask of dynamic driving tasks (DDT) that includes monitoring the driving environment and executing a response [1], [2]. OEDR is the key differentiation between Level-2 and Level-3+ autonomous driving, which enables tactical task automation by making control orders in response to special events and objects. On the other hand, these event data are valuable sources for the post-analysis of driving behaviors [3], corner cases [4], and perception failure scenarios [5]. It is necessary to log these event detection data onboard a vehicle.

Manuscript received XX November, 2022.

Corresponding author is Yinhai Wang (e-mail: yinhai@uw.edu).

Ruimin Ke is with the Department of Civil Engineering, The University of Texas at El Paso, El Paso, TX 79968 USA. Zhiyong Cui is with the School of Transportation Science and Technology, Beihang University, China. Yanlong Chen is with the Department of Mechanical Engineering, University of Tokyo, Bunkyo City, Tokyo, Japan. Meixin Zhu is with The Hong Kong University of Science and Technology (Guangzhou). Hao Yang, Yifan Zhuang, and Yinhai Wang are with the Department of Civil and Environmental Engineering, University of Washington, Seattle, WA 98195 USA.

This work was supported in part by the Safety Research and Demonstration (SRD) grant from the Federal Transit Administration (FTA) and in part by the Pacific Northwest Transportation Consortium (PacTrans).

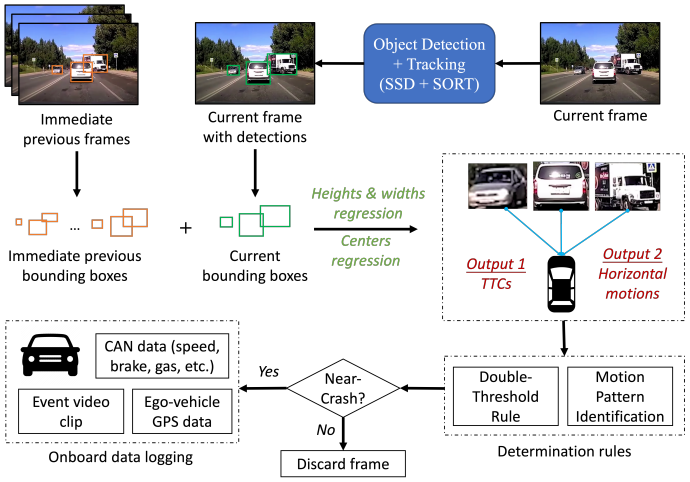


Fig. 1. The proposed overall methodology.

Vehicle event data logging is normally done through recording continuous data feeds locally on the device or remotely on the cloud [6], [7], [8]. Either way, it would require large storage and sometimes low transmission latency, but the fraction of useful data is small. For example, if an autonomous driving software tool collects sample data for training a Level-3 tactical lane-changing model, most data except lane changes would not be needed [9], [10]; a traffic safety project on studying wrong-way driving would only need those events where wrong-way driving occurs [11].

An onboard real-time event detection system is necessary to eliminate the majority of regular vehicle operation data and keeps those of interest. It is widely expected in applications such as vehicular federated learning [12] and traffic digital twins [13]. Nowadays, real-time onboard vehicle event detection is still in its infancy. The state-of-the-practice products on the market, such as the MobilEye Shield+ collision avoidance system [14] and the open-source Comma.ai autonomous driving toolkit [15], include limited OEDR functions and require the installation of extra sensors and devices with a relatively high price.

With the rapid emergence of edge computing, sensor data can be processed closer to where they are generated, thereby reducing computation, communication, and storage costs [16], [17]. Edge computing is ideal for advanced driving assistance systems (ADAS), and it is expected that OEDR and onboard

data logging will be supported by edge computing with high intelligence [18], [19]. It is also promising if existing sensors onboard a vehicle could be leveraged, which would facilitate the development of affordable and accessible ADAS as well as the large-scale deployment on existing vehicles.

Among all the events of interest to OEDR, quite some of them can be categorized into a group called near-crash. They would include any conflicts between the subject vehicle and other road users. The near-crash events are conflicts that could potentially develop into a collision. Herbert Heinrich discovered the relationship between major injury, minor injury, and no injury incidents (1 major injury incident to 29 minor injury incidents to 300 no injury incidents) [20]. This linear relationship still holds in traffic scenarios, though the exact ratios might be different [21], [22], [23], [24], [25], [26]. Near-crash has two major properties that make it valuable for a variety of research topics: (1) It reflects the underlying causes of the incidents while resulting in no or minor losses; (2) It is in a much larger number than real accidents to support meaningful model training and statistical analysis.

In this paper, we explore edge artificial intelligence in real-time vehicle event detection and logging. Specifically, the proposed system (see Fig. 1) targets vehicle-vehicle and vehicle-pedestrian near-crash events detection and real-time data logging. It uses existing camera sensors and communication services onboard transit buses. We develop a lightweight algorithm that models the bounding boxes of object detection and tracking output in linear complexity for time-to-collision calculation, and a few rules to determine if a near-crash event is of interest. The algorithm is not only efficient but also insensitive to camera parameters, such as focal length, so it can be conveniently transferred to another vehicle's onboard camera. We also add an additional event logging mechanism to record data from multiple onboard systems for individual events by sending triggers via the Controller Area Network (CAN); one system can receive the event triggers and record its data for the same event detected by the other system. This mechanism can be generalized to multiple onboard systems for event logging.

The contributions of this study are summarized below.

- A camera-parameter-free near-crash detection method on edge computing devices enabled by modeling the bounding boxes of deep learning object detection/tracking outputs. This method is derived in Section II.B and is a key component to enable real-time data processing and ensure high transferability to different onboard cameras.
- A parallel edge computing architecture with the video streaming thread, near-crash detection thread, data logging thread, and trigger thread. The architecture is described in Section II.A, which enables low latency in onboard video frame reading, dumping data of no interest, triggering event data collection and integration, and in support of the robustness of the main thread operations.
- An efficient mechanism for vehicle event logging from multiple data sources by communicating through CAN with other onboard internal or external systems. This mechanism expands the data diversity for an event of interest and has a high potential for advanced post-analysis.
- Extensive real-world testing and demonstration of the system were conducted on four transit buses for over a year in Pierce County, WA. The open-road testing of edge artificial intelligence systems for transit applications is among the first efforts and largest scale so far. The findings and lessons learned are valuable for future research.

"Lightweight" is claimed by (1) the linear-complexity modeling of bounding boxes on edge, (2) the parallel event trigger and logging mechanism that discards most of the redundant data onboard a vehicle, (3) quantization of the deep neural network in object detection, and (4) the real-time processing.

II. Methodology

A. Vehicle Event Data Integration and Logging Mechanism

The overall system architecture of the edge computing platform is shown in Fig. 2. Given the real-time operation requirement, the design is concise enough to be highly efficient and sophisticated enough for high accuracy and reliability. The near-crash detection method also should be insensitive to camera parameters to accommodate large-scale deployments.

The system is implemented in a parallel multi-thread manner. Parallel edge AI has been examined for multi-task transportation systems [27]. In this study, four different threads are operating simultaneously: the near-crash main thread, the event data logging thread, the video streaming thread, and the CAN trigger thread. The proposed near-crash detection method is implemented in the main thread. When near-crash events are detected, a trigger (trigger #1) will be sent to the data logging thread, and it will record video frames from a queue (a global variable) and other data that are associated with the near-crash event. The third thread for video streaming keeps the latest video frame captured from the camera in another queue and will dump previous frames when the capturing speed is faster than the main thread's frame processing speed. The CAN trigger thread provides additional information for each near-crash event with the ego-vehicle's speed, brake, acceleration, and so forth. Trigger #2 is optional, which is the detection trigger from another system through CAN, in case any other system is integrated for evaluation or joint detection.

The proposed mechanism ensures that the system delay is low. The video streaming thread ensures that the main thread reads the latest frame captured by the camera by not accumulating frames. The event data logging thread is designed as an individual thread to handle data transmission so that the main thread operation is not affected by the network bandwidth. The CAN trigger thread is for additional information collection, and the purpose for separating it as another individual thread is the consideration of system function extension. The proposed system can communicate with other systems via this thread while not affecting its performance.

B. Lightweight Near-crash Detection Algorithm

1) Understanding near-crash patterns in dashcam view: Relative motions between the ego-vehicle and other road users

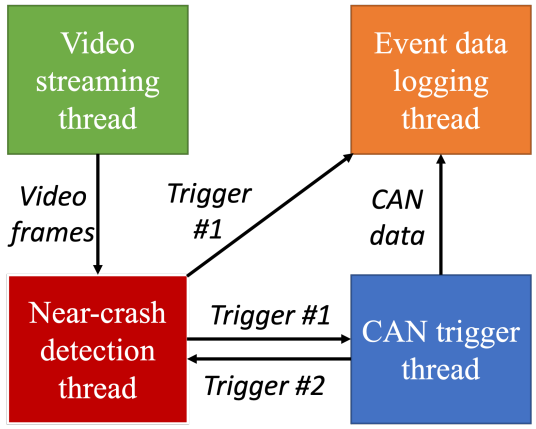


Fig. 2. The four parallel threads of the edge computing system.

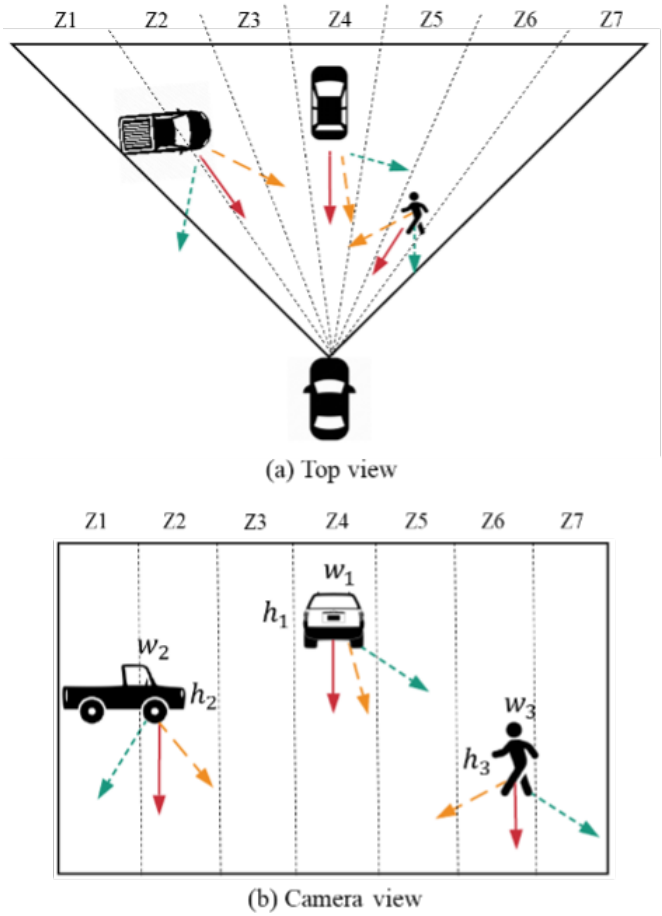


Fig. 3. Relative motion patterns in a dashcam view.

are important cues for near-crash detection using a single camera. Relative motion patterns as well as the relationship between a pattern in the camera view and its corresponding pattern in the real world must be understood (see Fig. 3). The relative motion patterns between two road users vary from case to case. Roadway geometry, road user's behavior, relative position, traffic scenario, etc. are all factors that may affect the relative motion patterns.

Relative motion that has the potential to develop into a crash /near-crash is characterized from the ego-vehicle's perspective as the target road user moving towards it. Regardless of the driver's behavior [28], this kind of relative motion is shown as a motion vector of the target road user moving vertically toward the bottom side of the camera view. Examples are shown as solid red arrows in Fig. 3. In the real-world top view, the three solid red arrows represent the relative motions between the ego-vehicle and each of the three road users (a pick-up truck, a car, and a pedestrian). Each of the three camera sight lines aligns with a relative motion vector (Z2, Z4, and Z7). In the camera view, the lines of sight are shown as vertical bands. The relative motion vectors for near-crashes in the top view correspond to vectors moving toward the bottom in the camera view aligning with Z2, Z4, and Z7.

In addition to the near-crash cases defined above, other patterns may occur. First, a target road user may move towards the ego-vehicle, move away from the ego-vehicle, or stay at the same distance to the ego-vehicle. These can be identified as object image size changes in the camera. This property will be utilized later in our approach. Image size decreasing or no size change would not indicate a potential crash or near-crash. For size increase, there are three cases. The first cases are the potential crashes, shown as the solid red arrows in Fig. 3. The second is the warning case, shown as the dotted orange arrows, in which the relative motion is towards the center line of sight of the camera (the pick-up truck and the pedestrian), or the relative motion is slightly different from the solid red arrow while the target road user is at the center line of sight (the car). The warning cases could develop into crashes if there are slight changes in the speeds or headings of either the target or the ego-vehicle. The third case is the safety case where relative motion is moving away from the center line of sight, shown as the dotted green arrows in Fig. 3.

2) Bounding boxes regression for TTC calculation: An object appears larger in the camera view as it is approaching the camera and smaller as the distance to the camera increases. Researchers at Mobileye published a paper as early as 2004 to show that it was possible to determine TTC using size changes [29]. In this study, the proposed approach for TTC estimation mainly considers: (1) leveraging the power of recent achievements in deep learning, (2) making the computation as efficient as possible to support real-time processing on Jetson, and (3) transferability to any dashboard camera without knowing the camera's intrinsic parameters.

The object detector and tracker adopted in the current system are the Single Shot Multibox Detector (SSD)-Inception net and the Simple Online Real-time Tracking (SORT) methods [30], [31]. These two blocks are generalizable to other methods. Object detection and tracking provide the locations, categories, and sizes of objects regarding the bounding boxes information. However, bounding boxes are approximate sizes of the objects and cannot be directly used for the accurate determination of object size. Particularly, given two consecutive frames, the size change of an object is subtle; and in many cases, this change is not recognizable due to noise in the bounding box generation.

Another reason for inaccurate size change detection in consecutive neighboring frames is that the time interval is too

small in between. Given a video with a frame rate of 24 FPS, the next frame is captured in less than 0.05 seconds. Thus, for size change detection, more frames are needed to compensate for the noise in each frame and increase the time interval for the detection. Linear regression is used for bounding boxes' heights or widths over a group of consecutive frames. The team found that 10 to 15 frames are enough to compensate for noise and the time associated with 10 to 15 frames is still small enough (about 0.5 seconds) to assume that the road user's motion is consistent.

Therefore, as shown in Fig. 1 again, the input to the linear regression is a list of heights or widths extracted from the bounding boxes, and the slope outputted by the regression will be the size change rate. Let us denote the change rate as r_t, and the size of the road user in the video frame as s_t at time t. At the same time, in the real world, the longitudinal distance between the target road user and the ego-vehicle is D_t, the relative longitudinal speed is V_t, the target road user's size is S_t, and the camera focal length is f. Based on the pinhole camera model, there is (1).

$$\frac{s_t}{f} = \frac{S_t}{D_t} \tag{1}$$

Relative speed is the first derivative of relative distance, and that size change rate is the first derivative of the object size over time, as shown in (2).

$$V_t = \frac{dD_t}{dt}, r_t = \frac{ds_t}{dt} \tag{2}$$

Since the real-world target road user's size does not change over time, there is the subsequent (3).

$$0 = \frac{dS_t}{dt} = \frac{d(\frac{D_t s_t}{f})}{dt} \tag{3}$$

And since the focal length does not change over time, we have (4).

$$0 = \frac{d(D_t s_t)}{dt} = \frac{dD_t}{dt} s_t + \frac{ds_t}{dt} D_t = V_t s_t + r_t D_t \tag{4}$$

Thus, the time-to-collision (TTC) can be estimated through (5) as the size of the bounding box at time t divided by the size change rate at time t. It is not related to the focal length or other intrinsic camera parameters. The TTC value can be either positive or negative, where being positive means the target is approaching the ego-vehicle, and being negative means it is moving away from the ego-vehicle.

$$TTC = -\frac{D_t}{V_t} = \frac{s_t}{r_t} \tag{5}$$

3) Height or width?: There are two options for the size of the road user in the camera view, height or width. We argue that height is a better indicator than width. From the ego-vehicle's perspective, it may observe a target vehicle's rear view, front view, side view, or a combination of them, depending on the angle between the two vehicles. That is to say, the bounding box's width change may be caused by either the relative distance change or the view angle change. For example, when the ego-vehicle is overtaking the target vehicle, or the target vehicle is making a turn, the view angle changes and will lead to the bounding box's width change.

However, the bounding box's height of the target vehicle is not influenced by the view angle; it is solely determined by the relative distance between the two vehicles. Similarly, a pedestrian walking or standing on the street may have different bounding box widths due to not only the relative distance to the ego-vehicle but also the pose of the pedestrian; but the height of a pedestrian is relatively constant.

Despite the challenge of using width to determine an accurate TTC, it still provides valuable information. Since we are using only less than one second of frames for the calculation, the view change does not contribute as much as the distance change, so width still roughly shows the longitudinal movement of the road user. This is very important in some cases. For instance, a vehicle moving in the opposite direction of the ego-vehicle is truncated by the video frame boundary. In this case, the height of the vehicle increases while the width decreases. This is not a near-crash case at all, but the TTC can be very small and falsely indicate a near-crash by only looking at the height change.

4) Double-threshold rule: We propose a double-threshold rule: if the TTC threshold for determining a near-crash is δ, we will set this δ as the TTC threshold associated with the height regression. At the same time, we have another TTC threshold φ associated with the width regression. The second threshold φ is to ensure that the width and height changes are in the same direction. The rule is represented as

$$0 < \frac{h}{r_h} < \delta, 0 < \frac{w}{r_w} < \varphi, \delta < \varphi \tag{6}$$

where r_h and r_w are the change rates for height h and width w. It is a necessary condition for a near-crash.

5) Horizontal motion estimation: As shown in Fig. 3, there are three scenarios for the case that a road user approaches the ego-vehicle; they correspond to potential crashes, warnings, and safe scenarios. Besides TTC, these scenarios can be differentiated with the relative horizontal motion between the ego-vehicle and the target. This needs to be calculated with computationally fast methods as well. We propose to apply another linear regression using a list of bounding box centers of the target road user. The regression result would be able to indicate the moving direction of the road user in the camera view.

In general, when the target's location is closer to the bottom and closer to the center line of sight, the risk of a collision is higher, so the threshold for the moving direction ω is looser. We propose a rule to show this judgment as

$$\alpha < \omega(C_x - C_{los})(B_y - B) < \beta \tag{7}$$

where C_x is the center's x coordinate, C_{los} is the center line of sight, B_y is the bottom side of the bounding box, and B is the bottom of the video frame. Since cameras have different resolutions, $(C_x - C_{los})$ is normalized to $[-1, 1]$ and $(B_y - B)$ is normalized to $[0, 1]$. The two thresholds are α and β; α should be set to negative to capture the potential warning scenarios (the orange dotted arrows in Fig. 3). And β should

be just slightly larger than zero to capture the potential crashes (the solid red arrows in Fig. 3) and filter out most of the safe scenarios (the green dotted arrows in Fig. 3). Equations (6) and (7) together identify near-crash events.

III. Experimental Results and Analysis

A. Experiment Design

Local experiments with locally stored videos at Jetson and real-world experiments with onboard real-time video feeds were selected as two groups for testing the system. Local video resources covered a lot of historical near-crash scenarios as well as other corner cases. It was a better source to evaluate the near-crash detection method we proposed in this paper. Real-time video stream data was captured by the system on cars and buses. Over 1000 hours of tests were conducted in the study. Local video data were also collected from online sources (e.g., YouTube) and dashboard cameras. Real-world tests have been conducted on four Pierce Transit buses for over a year in 2020 and 2021. Fig. 4 shows the system and testing buses for the real-world test. From top to bottom: the systems ready to be installed (before installation), three of the testing buses at Pierce Transit, the radio box behind the bus driver's seat where the system works, and the system being tested in the radio box.

Fig. 4. The system prototypes, buses for real-world testing, and the bus radio box where the system works.

B. System Hardware Components

The system consists of an Nvidia Jetson TX2 edge computer, a dashcam (can be a USB camera or IP camera), a GPS receiver, an in-vehicle power inverter, a PEAK CAN adapter for CAN bus communication, an external circuit based on Arduino board for auto bootup, a shell for the Jetson device, an ethernet cable, two power cables, an internet switch, mounting materials, and a cloud server. The Nvidia Jetson device is the key processing unit of the system, running near-crash detection, video streaming, data logging, CAN trigger threads, and algorithms. The Jetson was powered by in-vehicle (either car or bus) 12V DC power through the power inverter. The Arduino circuit is connected to the Jetson, and when the vehicle's power is on, it will auto-boot up the system.

C. Parameter Settings

Several key parameters needed to be set properly: SSD detector confidence threshold, the number of frames for size regression, the number of frames for center regression, TTC threshold δ, TTC threshold φ, horizontal motion threshold α, horizontal motion threshold β, and Jetson power mode. Given that the SSD detector tended to have fewer false positives than false negatives [32], some false positives can be filtered out at the tracking step, and more false positives (if any) will be filtered out by the near-crash detection algorithm, we set the detection confidence threshold to be 0.3–0.5.

For the number of frames for size regression, we suggested setting them to be around 10 to 15 frames. This range was large enough to compensate for the bounding box noises and small enough to assume the target's motion was consistent. The number of frames for center regression can be a little larger to capture the horizontal motion better, and the suggested number was in the range of 15 to 20. For δ and φ, as defined by many previous studies, the TTC threshold for a near-crash was around 2 to 3 seconds, which was our suggested value for δ. And we found that setting φ to about 2 to 2.5 times of δ worked well. We suggested setting α to the range of $[-1, -0.5]$ and β to $[0.02, 0.1]$. Jetson power mode

was recommended to be set as Max-N to fully utilize its computational power, though our system still operated in real-time (but with lower FPS) with Max-Q mode.

The parameters were tuned based on multiple rounds of local and real-world experiments. In the local tests, dashcam videos with the labels like *near-crash*, *traffic conflict* from different cameras were downloaded and used to fine-tune the parameters and thresholds to get a universally good performance on near-crash detection. In the real-world experiments, before deploying the devices on any of the project transit buses, two Honda cars were first connected with the system and driven around the university campus for preliminary real-world parameter settings. Later, the systems were tested on transit buses onsite at the Pierce Transit campus, with the buses in idle mode. We were able to adjust the network and power settings further as well as fine-tune the parameters, e.g., by having team members run toward the front of the bus to trigger the vehicle-pedestrian near-crash detection and data logging. The CAN communication function was first tested in the lab by connecting two PEAK CAN adapters through cables and sending CAN message to each other, and later the CAN thread was further fine-tuned on the buses via message exchange with another onboard system. During the open-road tests, the team actively monitored the system performance on the server side and regularly went onsite to examine the systems.

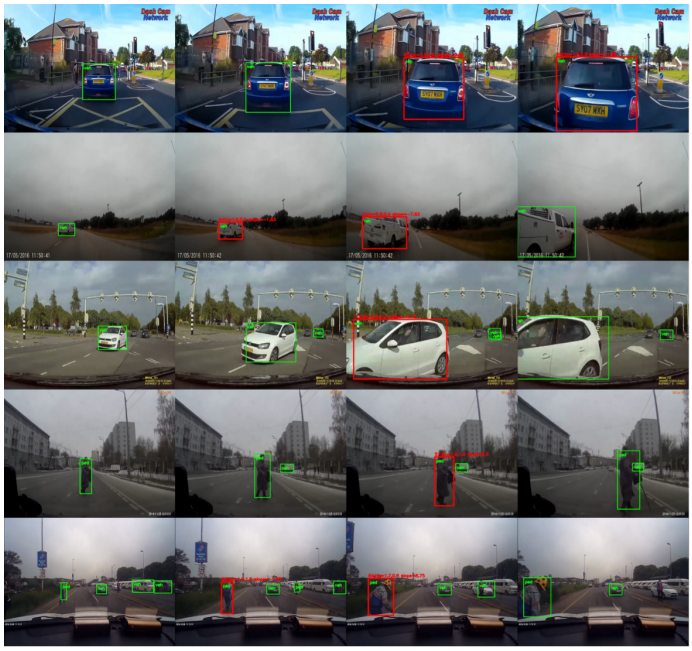


Fig. 5. Sample near-crash detection results, where red bounding boxes indicate the potential conflict with the road user. Each row is a four-frame sequence of one near-crash event.

D. Evaluation of Near-crash Detection

Essentially, near-crash is a type of traffic anomaly. To evaluate the proposed method's accuracy, we used the evaluation process of the Traffic Anomaly Detection task (Track 4) of the 2020 AI City Challenge as the reference [33]. First, the task dataset has 100 video clips with some anomalies. It is unknown exactly how many anomalies are in the test dataset, but the number is between 0 and 100, as mentioned in the introduction to Track 4. Likewise, we made a local test dataset with 5000 video clips with 500 near-crash events. As aforementioned, the test videos were from online resources and vehicle dashboard cameras. This dataset is not being published due to potential privacy and copyright issues. There is a plan to create such a video dataset for near-crash detection in the future.

We manually labeled all the near-crash events with their occurrence videos and times. Similar to the AI City Challenge Track 4, we defined a true-positive (TP) as a predicted near-crash within 10 seconds of the true near-crash. A false-positive (FP) is a predicted near-crash that is not a TP for a near-crash. A false-negative (FN) was a true near-crash that was not predicted. We used the F1 score to evaluate accuracy. F1 score was the harmonic mean of the precision and recall, where the best value = 1 and the worst value = 0.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \quad (8)$$

Sample near-crash detection results are shown in Fig. 5. The top three rows were three vehicle-vehicle near-crashes, and the bottom two rows were two vehicle-pedestrian near-crashes. The bounding boxes turned red to indicate a predicted near-crash, while other detected road users had green bounding boxes. A few more sample detection results can be found in the video published at https://www.youtube.com/watch?v=9NGo4Ef59i0.

The system correctly predicted 496 out of the 500 labeled near-crashes and missed just 4. It generated 8 FPs in the 5000 video clips. Based on 8, the final F1 score was 0.988, and the average processing speed with Max-N mode was about 18 FPS. The performance was promising, considering that we intentionally included a variety of near-crash scenarios and some very challenging cases in the dataset. There were adverse weather conditions (e.g., foggy, rainy, snowy), nighttime situations, traffic congestion, urban/rural traffic scenes, and so on. It is worth mentioning that the 5000 video clips are from a lot of different cameras and the proposed system knew nothing about the camera parameters of any of these cameras. This result benefited from the near-crash detection method. It again highlighted the possibility for low-cost and highly efficient large-scale application of the edge computing system to partially fulfill the purposes of safety data generation, corner case collection, and collision avoidance.

We carefully examined the FN and FP cases and summarized the causes. One of the four FNs that the system missed was a vehicle-pedestrian near-crash at night on a rural freeway with no streetlight. The pedestrian violated traffic rules by crossing the freeway, and the driver did not see him until almost ran into him. The pedestrian was entirely in the dark so the object detector missed him. Though there were more FPs than FNs, we considered only 8 FPs out of 5000 video clips acceptable and encouraging given the tradeoff in the efficiency of the system. While the proposed near-crash detection method can compensate for bounding box size noise in most cases, it was not perfect. In the fourth case (the fourth

row) of Fig. 5, right before the correct detection of this vehicle-pedestrian near-crash, there was a vehicle-vehicle FP caused by a significant error in vehicle size detection. It was included in our demo video.

E. Comparison and Discussion

This sub-section compares the proposed near-crash system qualitatively to the state-of-the-art on using dashboard cameras for near-crash detection. The comparison is presented in Table I. The state of the arts has fully automated the process of near-crash detection and data collection using regular computers, but this study is among the first efforts to adopt edge computing, design, and implement a system running on edge devices (Nvidia Jetson TX2). Regarding near-crash detection methods, the state of the arts tends to use machine learning, especially deep learning models. Convolutional neural network (CNN), long short-term memory (LSTM) neural network, and attention mechanism appear to be a good combination demonstrating superiority in some most recent studies [34], [35]. Ibrahim et al. also showed that a bi-directional LSTM with a self-attention mechanism performed better than single LSTM with regular attention [35]. However, these existing methods are more of black-box models due to the stack with multiple complicated deep learning modules thereby leading to limited efficiency, scalability, transferability, and interpretability.

It is okay in many projects just to leave the program running on regular computers and wait for the near-crash extraction to be done, but large-scale near-crash detection does require real-time processing to filter out irrelevant videos and other data as soon as possible to significantly save transmission bandwidth, disk storage, and post-processing time. The proposed system is among the first to achieve real-time near-crash detection with the designed algorithms, system architecture, and the concept of edge computing. The proposed method is not sensitive to camera parameters or labeled near-crash data, thus it has great transferability to different dashcams and a good chance to detect the types of corner cases not covered by the training dataset, which is often limited to a small scale in time and space.

The state of the arts was thoroughly validated with sufficient data, some have used thousands of video clips for validation purposes. In [36] and [34], the researchers used not only videos but also the telematics data such as acceleration and vehicle speed as part of the input, which indicated improved detection accuracy. In this study, we used online videos collected from different websites and unknown cameras for testing and finetuning the system and selected 5,000 video clips for validation. Then we deployed six of the devices on two cars and four buses. The system logged several hundred Gigabytes of vehicle-vehicle and vehicle-pedestrian near-crash videos and data, which were all filtered and transmitted to the cloud server in real-time.

The output data in most studies are videos, road user types, and risk levels associated with the events. Taccari et al. [36] estimated the TTC using a similar method to ours, but their estimation was based on solely two frames and without differentiation between using height or width of the bounding box since that was not their focus. The output of our method firstly includes TTC, road user type, and horizontal motion, and because of the real-time processing and CAN communication, it also collects the timestamp, latitude and longitude, speed, deceleration, brake switch, and throttle data. The accuracy is not directly comparable among the studies given the lack of a widely accepted benchmark dataset and the difference in processing unit, input and output data, and model specifications, but we list them in the table for reference.

In general, this study's prospective of application is very encouraging. It innovates in near-crash event logging by enabling real-time video analytics on the network edge and being backward compatible with existing vehicles. It addresses a few major concerns that are tied to some of the most critical research topics in intelligent vehicles and transportation, such as the lack of vulnerable road user safety data, the bottleneck of going large scale in corner case collection for AV testing, and bridging the gap between theory/simulation and practice in transportation.

F. System Transferability

The transferability of the system is excellent compared to the existing methods. First, the proposed system is designed to operate onboard vehicles with edge AI in the open road rather than taking videos as input in a lab-based environment. Second, the algorithm design is simple yet robust, with the bounding boxes modeling in a linear complexity and the TTC calculation not sensitive to camera parameters; that being said, the system has been demonstrated to operate on different vehicles with different camera settings. The key intrinsic camera parameter, i.e., the focal length, is canceled out in the bounding box regression derivation. Third, the road user detection and tracking methods, in this case, SSD and SORT, can be replaced by other similar detection and tracking methods, and the overall data flow within the system will remain the same. The realization of lightweight vehicular edge AI is also the foundation of vehicle crowdsensing intelligence towards decentralized salable automated vehicle services [37].

G. Edge-based Event Logging

While in the local test, Jetson processed the local videos frame by frame; in the real-world test, different camera hardware, settings, and different software design resulted in different frame-reading speeds and stability. This was why the video reading function was designed as an individual thread. Also, when doing the bounding box size regressions, the system included the corresponding time for each value (height, width, and center) because the intervals between each pair of neighboring frames may not be uniform. Moreover, the camera type may influence system performance. About 2s latency in the video feed on the bus was noticed due to the use of an IP camera connected via ethernet cables to the edge computing system. Jetson TX2 does not support auto boot-up. An external circuit driven by an Arduino board was developed to automatically boot up the system.

In the field test, approximately 6GB to 7GB of data were logged for one transit vehicle in an average month. The logged

TABLE I
COMPARISON WITH THE STATE OF THE ARTS

Research work	Ke et al. 2017 [26]	Kataoka et al. 2018 [25]	Taccari et al. 2018 [36]	Yamamoto et al. 2020 [34]	Ibrahim et al. 2021 [35]	**This study**
processing unit	Regular computer	Regular computer with GPU	Regular computer with GPU	Regular computer with GPU	Regular computer with GPU	**Nvidia Jetson TX2**
Edge computing	No	No	No	No	No	**Yes**
Key methods	HOG, SVM, Optical Flow	Two-stream CNN, Semantic Flow	YOLO3, Optical Flow, Random Forest	CNN, LSTM, Attention	CNN, Bi-LSTM, Self-Attention	**Bounding Box Modeling, SSD, SORT**
Real-time processing	No	No	No	No	No	**Yes**
Camera calibration or labeled data	Yes	Yes	Yes	Yes	Yes	**No**
Experimental data	30 hours of video	6,200 video clips	SHRP 2 data with videos and telematics data	4,200 video clips (15s each) and telematics data	74,477 sequential frames	**Online videos, two cars, four buses; 5,000 video clips for validation and over 10,000 hours of real-world testing**
output near-crash data	Pedestrian-related near-crashes	Risk level (high or low road user type)	Risk level (crash, near-crash, safe event), TTC, road user type	Near-crash type in five risk levels, road user type	Near-crash label	**TTC, road user type, horizontal motion, timestamp, event location, vehicle trajectory, speed, deceleration, brake switch, throttle**
Accuracy	0.900	0.645	0.870	Confusion matrix	0.994	**0.988**

data volume was less than 3 percent of the total data volume. The proposed system significantly reduced the redundant data onboard. Fig. 6 displayed the near-crash events GPS locations in October 2020 logged on Pierce Transit Bus #232; they were recorded into vehicle-pedestrian and vehicle-vehicle near-crashes. Fig. 7 presented two sample events logged on May 7th, 2021. Video clips as well as other associated data collected through the edge trigger mechanism (e.g., deceleration, speed, and brake switch status) were aggregated for the events. In the first sample event, the ego-bus was approaching a stopped silver car in front with a high deceleration. The second sample event occurred when the ego-bus was approaching a person standing at a bus stop; the bus decelerated with a steering angle to the right, resulting in a near-crash event.

IV. CONCLUSION

In this paper, we introduced the motivation, design, development, and evaluation of a lightweight edge computing system for real-time near-crash detection and event logging. The proposed system was driven by real-time video analytics on edge computing devices using existing dashcams. With the designs system-wise and algorithm-wise, this paper addressed several key challenges in vehicle near-crash detection and edge intelligence-based mechanisms for event logging. Thorough real-world experiments and analyses were conducted on cars and buses. The results were promising, demonstrating the potential of the proposed system for large-scale deployment with advantages including low cost, real-time processing, high accuracy, and great compatibility with different vehicles and cameras. The system can filter out events of no interest onboard a vehicle, largely saving network and computing resources. It also increased the output data diversity; the data was expected to be very valuable sources for intelligent vehicle and transportation applications, such as by serving as surrogate data for traffic safety studies and as corner case data for automated vehicle testing research. The system architecture is transferable to other OEDR applications, onboard camera settings, and the usage of other object detection/tracking algorithms. Future work will be focused on two aspects. The first aspect is that the evaluation metrics can be further expanded. Data consistency and the edge computer under different network environments are the metrics that could also be useful to similar system evaluations and studies. The second aspect is that we plan to investigate the detection and logging of other events of interest in the field of intelligent vehicles, such as pedestrian behavior prediction [38], driver intention recognition [39], and vehicle pose estimation [40].

ACKNOWLEDGMENTS

The authors would like to thank the Federal Transit Administration (FTA) and the Pacific Northwest Transportation Consortium (PacTrans) for funding this research. They express gratitude to their research partners (Pierce Transit, WSTIP, DCS Technology Inc., VTTI, Volpe Lab, CUTR, Veritas, Dr. Jerome Lutin, Ms. Janet Gates, etc.) in the FTA project team for their invaluable contributions.

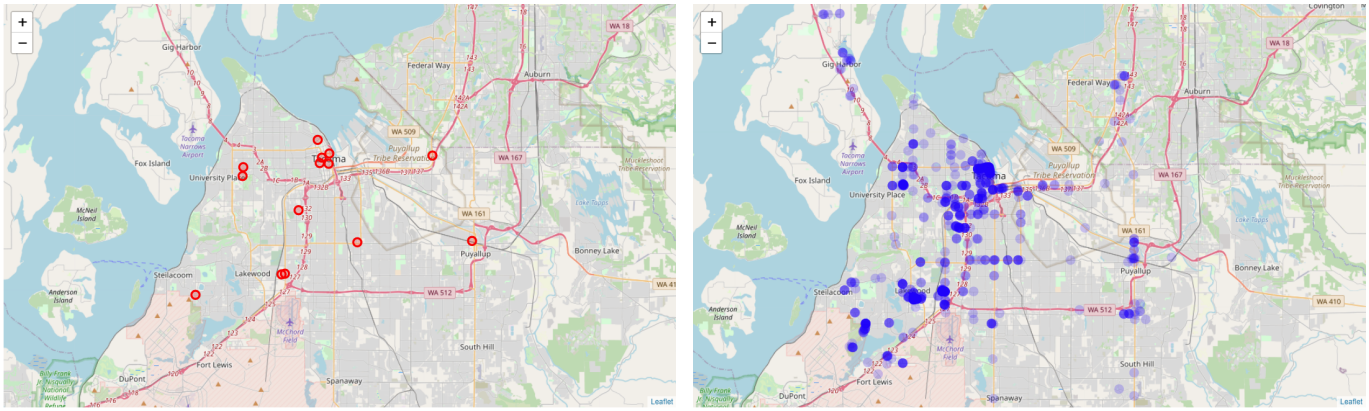


Fig. 6. Logged events locations of Bus 232 in October 2020; vehicle-pedestrian near-crashes (left) and vehicle-vehicle near-crashes (right). Transparency represents the different densities of events near the same location.

Fig. 7. Sample event data logged from a vehicle-vehicle near-crash (left) and a vehicle-pedestrian near-crash (right) on May 7th, 2021. Horizontal axes are the time axes in all sub-figures.

REFERENCES

[1] P. Koopman and F. Fratrik, "How many operational design domains, objects, and events?" *Safeai@ aaai*, vol. 4, 2019.

[2] E. Thorn, S. C. Kimmel, M. Chaka, B. A. Hamilton *et al.*, "A framework for automated driving system testable cases and scenarios," United States. Department of Transportation. National Highway Traffic Safety Administration, Tech. Rep., 2018.

[3] Z. Hu, Y. Xing, W. Gu, D. Cao, and C. Lv, "Driver anomaly quantification for intelligent vehicles: A contrastive learning approach with representation clustering," *IEEE Transactions on Intelligent Vehicles*, 2022.

[4] D. Bogdoll, J. Breitenstein, F. Heidecker, M. Bieshaar, B. Sick, T. Fingscheidt, and M. Zöllner, "Description of corner cases in automated driving: Goals and challenges," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1023–1028.

[5] S. Safavi, M. A. Safavi, H. Hamid, and S. Fallah, "Multi-sensor fault detection, identification, isolation and health forecasting for autonomous vehicles," *Sensors*, vol. 21, no. 7, p. 2547, 2021.

[6] J. Lutin, H. Soule, D. Sellers, D. Valadez, J. Sellers, Y. Wang, A. Krum, M. Golusky, L. Fischer, R. Ke *et al.*, "Pierce transit automated collision avoidance and mitigation safety research and demonstration project, final report," United States. Department of Transportation. Federal Transit Administration, Tech. Rep., 2022.

[7] Z. Zhou, F. Zhu, D. Xu, B. Chen, S. Guo, and Y. Dai, "Event-triggered multi-lane fusion control for 2-d vehicle platoon systems with distance constraints," *IEEE Transactions on Intelligent Vehicles*, 2022.

[8] Z. Zhong, L. Zhao, B. Dimitrijevic, D. Besenski, and J. Lee, "Assessing connected vehicle data coverage on new jersey roadways," *arXiv preprint arXiv:2208.04703*, 2022.

[9] G. Li, Y. Qiu, Y. Yang, Z. Li, S. Li, W. Chu, P. Green, and S. E. Li, "Lane change strategies for autonomous vehicles: a deep reinforcement learning approach based on transformer," *IEEE Transactions on Intelligent Vehicles*, 2022.

[10] Y. Liu, Z. Wang, K. Han, Z. Shou, P. Tiwari, and J. Hansen, "Vision-cloud data fusion for adas: A lane change prediction case study," *IEEE Transactions on Intelligent Vehicles*, 2021.

[11] Z. Rahman, A. M. Ami, and M. A. Ullah, "A real-time wrong-way vehicle detection based on yolo and centroid tracking," in *2020 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2020, pp. 916–920.

[12] Y. Tian, J. Wang, Y. Wang, C. Zhao, F. Yao, and X. Wang, "Federated vehicular transformers and their federations: Privacy-preserving computing and cooperation for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2022.

[13] Z. Wang, K. Han, and P. Tiwari, "Digital twin-assisted cooperative driving at non-signalized intersections," *IEEE Transactions on Intelligent Vehicles*, 2021.

[14] J. Spears, J. Lutin, Y. Wang, R. Ke, and S. M. Clancy, "Active safety-collision warning pilot in washington state," Tech. Rep., 2017.

[15] G. Haciane, R. Lerdphayakkarat, P. Meteekotchadet, J. Kutch, J. Roschke, and H. Kutgun, "Comma. ai marketing plan," 2018.

[16] H. F. Yang, J. Cai, C. Liu, R. Ke, and Y. Wang, "Cooperative multi-camera vehicle tracking and traffic surveillance with edge artificial intelligence and representation learning," *Transportation Research Part C: Emerging Technologies*, vol. 148, p. 103982, 2023.

[17] X. Zhou, R. Ke, H. Yang, and C. Liu, "When intelligent transportation systems sensing meets edge computing: Vision and challenges," *Applied Sciences*, vol. 11, no. 20, p. 9680, 2021.

[18] 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.

[19] F.-Y. Wang, "Metavehicles in the metaverse: Moving to a new phase for intelligent vehicles and smart mobility," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 1–5, 2022.

[20] H. W. Heinrich *et al.*, "Industrial accident prevention. a scientific approach." *Industrial Accident Prevention. A Scientific Approach.*, no. Second Edition, 1941.

[21] C. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, D. J. Ramsey *et al.*, "The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data," 2006.

[22] J. Wu, H. Xu, Y. Zheng, and Z. Tian, "A novel method of vehicle-pedestrian near-crash identification with roadside lidar data," *Accident Analysis & Prevention*, vol. 121, pp. 238–249, 2018.

[23] A. Talebpour, H. S. Mahmassani, F. Mete, and S. H. Hamdar, "Near-crash identification in a connected vehicle environment," *Transportation Research Record*, vol. 2424, no. 1, pp. 20–28, 2014.

[24] H. Makizako, H. Shimada, R. Hotta, T. Doi, K. Tsutsumimoto, S. Nakakubo, and K. Makino, "Associations of near-miss traffic incidents with attention and executive function among older japanese drivers," *Gerontology*, vol. 64, pp. 495–502, 2018.

[25] H. Kataoka, T. Suzuki, S. Oikawa, Y. Matsui, and Y. Satoh, "Drive video analysis for the detection of traffic near-miss incidents," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3421–3428.

[26] R. Ke, J. Lutin, J. Spears, and Y. Wang, "A cost-effective framework for automated vehicle-pedestrian near-miss detection through onboard monocular vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 25–32.

[27] R. Ke, C. Liu, H. Yang, W. Sun, and Y. Wang, "Real-time traffic and road surveillance with parallel edge intelligence," *IEEE Journal of Radio Frequency Identification*, 2022.

[28] Z. Wang, X. Liao, C. Wang, D. Oswald, G. Wu, K. Boriboonsomsin, M. J. Barth, K. Han, B. Kim, and P. Tiwari, "Driver behavior modeling using game engine and real vehicle: A learning-based approach," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 738–749, 2020.

[29] E. Dagan, O. Mano, G. P. Stein, and A. Shashua, "Forward collision warning with a single camera," in *IEEE Intelligent Vehicles Symposium, 2004.* IEEE, 2004, pp. 37–42.

[30] C. Ning, H. Zhou, Y. Song, and J. Tang, "Inception single shot multibox detector for object detection," in *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 549–554.

[31] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[32] R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, "A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on iot devices," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4962–4974, 2020.

[33] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, Y. Yao, L. Zheng, P. Chakraborty, C. E. Lopez *et al.*, "The 5th ai city challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4263–4273.

[34] S. Yamamoto, T. Kurashima, and H. Toda, "Identifying near-miss traffic incidents in event recorder data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2020, pp. 717–728.

[35] M. R. Ibrahim, J. Haworth, N. Christie, and T. Cheng, "Cyclingnet: Detecting cycling near misses from video streams in complex urban scenes with deep learning," *IET Intelligent Transport Systems*, vol. 15, no. 10, pp. 1331–1344, 2021.

[36] L. Taccari, F. Sambo, L. Bravi, S. Salti, L. Sarti, M. Simoncini, and A. Lori, "Classification of crash and near-crash events from dashcam videos and telematics," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2460–2465.

[37] Z. Zhu, X. Wang, Y. Zhao, S. Qiu, Z. Liu, B. Chen, and F.-Y. Wang, "Crowdsensing intelligence by decentralized autonomous vehicles organizations and operations," *IEEE Transactions on Intelligent Vehicles*, 2022.

[38] D. Yang, H. Zhang, E. Yurtsever, K. A. Redmill, and Ü. Özgüner, "Predicting pedestrian crossing intention with feature fusion and spatio-temporal attention," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 2, pp. 221–230, 2022.

[39] C. Wang, F. Li, Y. Wang, and J. R. Wagner, "Haptic assistive control with learning-based driver intent recognition for semi-autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, 2021.

[40] C. Zhao, C. Fu, J. M. Dolan, and J. Wang, "L-shape fitting-based vehicle pose estimation and tracking using 3d-lidar," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 787–798, 2021.

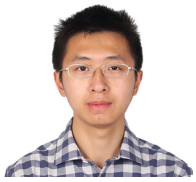
Ruimin Ke (Member, IEEE) received the B.E. degree in automation from Tsinghua University in 2014, the master's and Ph.D. degrees in civil engineering (transportation) from the University of Washington in 2016 and 2020, respectively, and the M.S. degree in computer science from the University of Illinois Urbana–Champaign. He is currently an Assistant Professor with the Department of Civil Engineering, The University of Texas at El Paso. His research interests include intelligent transportation systems and smart cities with a focus on video image processing, machine learning, and the Internet of Things applications. He is a recipient of the COTA Best Dissertation Award 2020–2021 and the TRB Best Paper Award (AED30) 2023.

Zhiyong Cui is an Associate Professor in the School of Transportation Science and Engineering at Beihang University, China. He was a University of Washington (UW) Data Science Postdoctoral Fellow at the eScience Institute. He received a B.S. degree in software engineering from Beihang University in 2012, an M.S. degree in software engineering and microelectronics from Peking University in 2015, and his Ph.D. in civil engineering at UW in 2021. He is the winner of the IEEE ITSS Best Dissertation Award in 2021 and the Best Paper Award at the 2020 IEEE International Smart Cities Conference. His primary research focuses on deep learning, machine learning, urban computing, traffic forecasting, connected vehicles, and transportation data science.

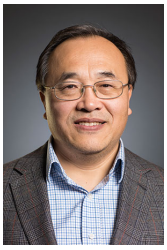
Yanlong Chen received the B.E. degree in mechanical engineering from Tsinghua University in 2019 and an M.E. degree in mechanical engineering from the University of Tokyo in 2022. In 2019, he was a visiting student in the Smart Transportation Application and Research Laboratory (STAR Lab) at the University of Washington. His research interests include computer vision and its application in robot manipulation.

Meixin Zhu is a tenure-track Assistant Professor in the Thrust of Intelligent Transportation (INTR) under the Systems Hub at the Hong Kong University of Science and Technology (Guangzhou). He is also an affiliated Assistant Professor in the Civil and Environmental Engineering Department at the Hong Kong University of Science and Technology. He obtained a Ph.D. degree in intelligent transportation at the University of Washington (UW) in 2022. He received his BS and MS degrees in traffic engineering in 2015 and 2018, respectively, from Tongji University. His research interests include Autonomous Driving Decision Making and Planning, Driving Behavior Modeling, Traffic-Flow Modeling and Simulation, Traffic Signal Control, and (Multi-Agent) Reinforcement Learning. He is a recipient of the TRB Best Dissertation Award (AED50) in 2023.

Hao Yang (Graduate Student Member, IEEE) received the B.S. degree of Telecommunication Engineering from both Beijing University of Posts and Telecommunications (2017) and University of London (2017). He is currently a Ph.D. student at the Smart Transportation Research and Application Lab (STAR Lab), Department of Civil and Environmental Engineering, University of Washington. He is an associate editor for IEEE ITSC 2021, reviewer of CVPR, IEEE Trans. on ITS, IEEE Trans. on NNLS and etc. His research interests are focused on computer vision, edge computing and transportation data analysis. He is a recipient of the TRB Best Paper Award (AED30) 2023.



Yifan Zhuang received his B.E. degree in Automation from Tsinghua University, Beijing (2016). Then he received an M.S. (2019) and a Ph.D. (2022) degree in Civil and Environmental Engineering from the University of Washington, Seattle. Dr. Zhuang worked as a Research Assistant at the Smart Transportation Application and Research (STAR) Lab from 2016 to 2022. Besides academic research, Dr. Zhuang has multiple experiences in industrial fields, including Horizon Robotics, Tencent, Adobe, and Google. His primary research interests include the application of edge computing in the transportation field, traffic sensing technologies, and computer vision in smart transportation facilities and autonomous driving. Dr. Zhuang is an active researcher in the intelligent transportation and computer science fields and has produced impressive work focusing on vision-based object detection and hardware integration. His cross-expertise in software algorithms and electronic hardware makes him solve problems more efficiently from a higher perspective. His research work generated significant impacts through a practical implementation for pedestrian safety and was reported by Seattle local news.



Yinhai Wang (Fellow, IEEE) received the master's degree in computer science from the University of Washington (UW) and the Ph.D. degree in transportation engineering from the University of Tokyo, in 1998. He is currently a Professor in transportation engineering and the Founding Director of the Smart Transportation Applications and Research Laboratory (STAR Lab), UW. He also serves as the Director of the Pacific Northwest Transportation Consortium (PacTrans), USDOT University Transportation Center for Federal Region 10, and the Director of the Northwestern Tribal Technical Assistance Program (NW TTAP) Center. His active research fields include traffic sensing, urban mobility, e-science of transportation, and transportation safety. He serves as the Chair of the Artificial Intelligence and Advanced Computing Committee of the Transportation Research Board. He is a fellow of the American Society of Civil Engineers (ASCE) and the Past President of the ASCE Transportation and Development Institute (T&DI). He is also a member of the IEEE Smart Cities Technical Activities Committee and was an elected member of the Board of Governors for the IEEE ITS Society from 2010 to 2013. He was the winner of the ASCE Journal of Transportation Engineering Best Paper Award for 2003 and the Institute of Transportation Engineers (ITE) Innovation in Education Award for 2018. He is also an Associate Editor of three journals, the Journal of ITS, PLOS One, and the Journal of Transportation Engineering.