Raven: Vision-Based Connected Vehicle Safety Platform Using Infrastructure Sensing, 5G, and MEC

Kamran Ali, Bo Yu, Vivek Vijaya Kumar, Krishnan Hariharan, *Fellow, IEEE*, Fan Bai, *Fellow, IEEE*, Mohammad Rehan, and Abhishek Vijay Kumar

***Abstract*—5G cellular communication and Multi-access Edge Computing (MEC) have provided a new opportunity to support connected vehicles safety applications complementing Dedicated Short Range Communication (DSRC) and Cellular Vehicle-to-Everything (C-V2X) direct communication. One major benefit of using cellular networks for connected safety is its wide deployment and support in the vehicles today, and it eliminates the need for additional physical radio technologies in vehicles. However, a few major considerations for cellular networks to support real-time safety applications are latency, reliability of communication, cellular costs, and scalability. In this paper, we conduct the first real-world implementation and study of using infrastructure-based camera sensing along with 5G cellular communication and MEC computing setup to support connected safety applications. To achieve this, we design and implement a vision-based multilayer connected safety platform called *Raven*. We also develop and evaluate three different connected safety applications namely Intersection T-Bone Vehicle Crash Warning, Vehicle-in-blind-spot Warning and Stopped/Disabled Hazard Vehicle Warning applications on top of the *Raven* platform. These applications are enabled via a roadside camera that is connected via 5G communication link to the MEC server where *Raven* detects and tracks vehicles in the received camera frames, estimates each vehicle's dynamics such as tracking ID, position, speed, and heading, packages that information along with other parameters required by the Basic Safety Messages (BSM) standard, and shares it with other vehicles that have subscribed to *Raven*'s service. Our experiments show promising results: the median error in position, velocity, and heading accuracy is 2.5 m, 4.9 kph (or 3 mph), and 2.1°, respectively, and the driver is warned in-time during all the test runs of the implemented safety applications.**

***Index Terms*—Connected vehicles, active safety, situational awareness, multi-access edge computing (MEC), V2X/V2N, 5G.**

I. Introduction

EXISTING advanced driver assistance systems (ADAS) based on object detection sensors installed on a standalone vehicle only may not adequately meet the safety requirements of driving scenarios that require non-line-of-sight (NLOS) knowledge of the road users who maybe occluded from LOS of the ego vehicle, e.g., during an intersection T-bone vehicle crash. Vehicle-to-everything (V2X) connectivity technologies such as dedicated short-range communications (DSRC) [1] and Cellular V2X (C-V2X) [2] allow effective sharing of information among road users and have shown the potential to significantly improve driving safety in such scenarios and can greatly reduce the possibility of collisions. Leveraging cameras installed on roadside infrastructure to track mobility of vehicles and vulnerable road users (VRUs) on a road and sharing that information among nearby vehicles and road users is one such application [3], [4].

Traditionally, V2X based safety application services have been envisioned with direct communication between vehicles or between vehicles and infrastructure for the most critical services, whereas connections through the cellular network (V2N) are usually envisioned for infotainment, telematics, or traffic management services. However, the wide-spread deployment of V2X technologies has been progressing at a moderate pace, while significant resources have been put into the research, development, and standardization of the technologies and systems in the past two decades. Some of the possible reasons are as follows: 1) Lack of government mandate [5]. This means vehicle manufacturers need to voluntarily integrate V2X technologies into production vehicles. Absence of a clear mandate does not guarantee that vehicles from one manufacturer will be able to communicate with vehicles from other manufacturers. 2) Lack of business models. A commercial ecosystem among vehicle manufacturers, service providers, application developers, and consumers is needed to successfully commercialize a new technology. 3) Wireless technology evolution. A new radio technology, such as C-V2X, replaces an old technology, such as DSRC, before the old technology gets a chance to be commercialized. C-V2X and DSRC are not compatible with each other, and messages can't be exchanged between C-V2X and DSRC radios. However, the capabilities introduced by 5G NR have raised expectations on the possibility of using the cellular link (e.g. the Uu link in 5G) to support connected safety using V2N or V2N2V communications [6], [7], [8], which has the potential to help avoid the aforementioned roadblocks by complementing the V2X direct communication efforts. Moreover, Multi-access Edge Computing (MEC) [9], [10], [11] promises reduced latency and near real-time performance to high-bandwidth applications by moving compute resources to the edge of the network. These compute resources can enable real time processing of data from existing infrastructure cameras to support computer vision based connected safety applications having end-to-end latency requirements of under 150 ms [8], [11], [12], [13], [14].

Manuscript received 16 December 2023; revised 29 April 2024; accepted 24 June 2024. Date of publication 19 July 2024; date of current version 7 November 2024. The review of this article was coordinated by Dr. Hong Wang. *(Corresponding author: Kamran Ali.)*

Kamran Ali, Bo Yu, Vivek Vijaya Kumar, Krishnan Hariharan, and Fan Bai are with the General Motors R&D, Warren, MI 48092 USA (e-mail: kamran.1.ali@gm.com; bo.3.yu@gm.com; vivek.vijayakumar@gm.com; hariharan.krishnan@gm.com; fan.bai@gm.com).

Mohammad Rehan is with the Cognizant, Rochester, MI, 48034 USA (e-mail: muhammad.rehan@cognizant.com).

Abhishek Vijay Kumar is with the Danlaw Inc., Novi, MI, 48375 USA (e-mail: abhishekk@danlawinc.com).

Digital Object Identifier 10.1109/TVT.2024.3430815

Motivation: Our work is motivated by the following questions: *Is it feasible to reliably enable roadside infrastructure-supported computer vision-based connected safety applications over a 5G cellular IP-based communication link such that the video data is wirelessly transmitted from an infrastructure camera to an edge server where a real-time computer vision algorithm detects, tracks, and localizes various different objects in LOS of the camera and provides this information to the nearby subscribed road users?* A positive answer would pave the way towards converting existing intersections into smart intersections in a relatively much shorter time frame compared to installing specialized radios and expensive computing hardware that may take decades. Although several works have discussed connected safety and edge computing in the context of enabling latency sensitive connected vehicle applications [6], [11], [12], [15], [16], [17], [18], [19], [20], [21], [22], yet to the best of authors' knowledge there is no published work on E2E implementation and systematic evaluation of vision-based connected safety applications using a real-world 5G MEC setup deployed by a commercial service provider.

Implementation: To this end, we design and implement such an infrastructure-supported vision-based connected safety platform, named *Raven*, and test it over a 5G cellular communication link via a private 5G tower with MEC-based high-end compute capability. In terms of safety applications, we implement and test Intersection T-Bone Vehicle Crash Warning, Vehicle-in-Blind-Spot Warning, and Stopped/Disabled Vehicle Warning applications on our *Raven* platform. Information flows through *Raven*'s pipeline in three key steps. First, video frames are streamed from the roadway infrastructure camera to the MEC server. At the server side, *Raven* detects and tracks important roadside objects (e.g., vehicles and pedestrians) in the received frames, estimates each object's dynamics such as tracking ID, position, speed, heading, etc., packetizes that information, and shares it with the vehicles that have subscribed to *Raven*'s service. Finally, the receiving vehicles extract information about the detected objects and put them into the standard Basic Safety Messages (BSM) [23] format. These BSMs are used for threat assessment and generating potential vehicle crash safety warnings.

To support accurate detection, localization and tracking of vehicles and other objects in the camera frames while keeping latency and bandwidth consumption low, we propose several useful optimizations to *Raven*. First, we employ a real-time *image super-resolution* technique [24] that allows us to reduce the resolution of video frames streamed by the infrastructure camera by leveraging a low GPU footprint machine learning model running at the edge server to upsample the received frames before feeding them to object detection and tracking module based on YOLO [25]. Second, we propose an efficient *neighborhood-based non-max suppression* (NB-NMS) algorithm on top of YOLO's standard NMS to filter spurious YOLO detections where the same object is detected multiple times but with different class labels instead. The standard NMS algorithm runs on class specific bounding boxes only, i.e., it does not take into account that same object can be detected as multiple different objects, e.g., a car detected as both car and a truck with slightly different bounding boxes. This leads to generation of spurious safety messages which is an undesirable situation. Our NB-NMS algorithm essentially performs NMS on all bounding boxes centered within a threshold pixel radius of each other independent of their class labels such that the computational complexity is constrained within that radius. Third, instead of using compute intensive Neural Networks (NN)-based or Optical Flow-based trackers, we propose a computationally simpler application specific *spatio-temporal tracker* to assign tracking IDs in a spatially and temporally consistent manner by looking for matches in frame history and employing a temporal voting strategy. Fourth, we propose an Unscented Kalman Filter (UKF) based tracking of vehicle dynamics (e.g. position and velocity) in world-coordinates. Selection of UKF instead of the standard KF or Extended KF (EKF) helps address non-linearity involved in converting points from image to world-coordinate frame. Fifth, we develop a *relevance priority* based message queuing strategy such that the information about the most relevant objects is transmitted first from MEC server to vehicles. Here the priority can be defined based on regions of interest (ROI), distance, speed, heading, size, and type of the detected object. We integrate this strategy with *Raven*'s ZeroMQ (ZMQ)-based publish/subscribe (PUB/SUB) messaging interface. This increases the chances of higher priority messages delivered in time before they are dropped as MEC messages remain in queue only for a small amount of time and then dropped as they become obsolete quickly.

Summary of Experimental Results: The results of our experiments conducted in real-world empirical environments are promising. Specifically, the median end-to-end latency recorded was 103 ms, indicating a fast and efficient real-time system. Additionally, we observed a median error of 2.5 m, 4.9 kph (or 3 mph), and 2.1° for position, velocity, and heading accuracy, respectively. *Raven*'s reliable range of operation is approximately 80 m from the infrastructure camera. During all the test runs of the implemented safety applications, the driver was alerted in a timely manner indicating that the applications work effectively and the accuracy and precision of the sensing, localization and tracking pipeline is adequate.

Key Contributions: The key contributions of this work are summarized as follows:

1) Designing and implementing an E2E infrastructure-supported vision-based connected safety platform to enable vehicular safety applications. We implemented and tested our system via a real-world 5G cellular MEC setup deployed by a commercial service provider.
2) Proposing an application specific, practical, and efficient mono-camera based detection, localization, and tracking solution capable of providing lane level or better lateral position tracking at up to a longitudinal distance of 80 meters, while keeping the E2E latency low via various strategies such as image super-resolution and object relevance priority based message queue.
3) Evaluation of three important connected safety applications in real experimental vehicles, i.e., Intersection T-Bone Vehicle Crash Warning, Vehicle-in-Blind-Spot Warning, and Stopped/Disabled Vehicle Warning applications by implementing them on top of our proposed *Raven* platform. Our system is flexible and can be extended to support other related roadside connected safety applications such as Pedestrian-at-Crosswalk Warning.

II. Related Work

Connected Safety Communication over Cellular Networks: Several prior works have discussed the possibility of supporting connected safety services using V2N or V2N2V communications [6], [7], [8], [26]. Coll-Perales et al. introduce an E2E latency model to quantify the latency of 5G V2N and V2N2V communications. Using this model, they estimate the E2E latency of 5G V2N2V connections for a various possible 5G network deployments and configurations. Kanavos et al. investigate the extent of support that 4G and 5G networks can offer to connectivity use cases in terms of delay and spectrum needs in [8]. Martin et al. evaluate the performance of different connectivity alternatives (based on LTE-Advanced) for the support of connected services in a simulated highway scenario [26]. To the best of the authors' knowledge, all such prior works are either based on simulation and/or mathematical analysis based on latency modeling. In contrast, our work is based on implementation of E2E vision-based connected safety applications that we evaluate using a real-world 5G cellular tower and MEC server setup deployed by a commercial service provider. Although a similar concept has been demonstrated in an online video [27], yet no details of the implementation and performance are available.

Computation Offloading and Edge Computing (MEC): Several works have discussed cellular network communication and edge computing in context of enabling latency sensitive connected vehicle safety applications [6], [12], [15], [16], [17], [18], [19], [20], [21], [22], [28]. Giust et al. [12] and Hung et al. [28] showcase the automotive use cases that are relevant for MEC, providing insights into the technologies specified and investigated by the ETSI Industry Specification Group MEC. Emara et al. [6] simulate a VRU alert use case assuming 5G Uu radio interface and argue that stringent latency requirements posed by the connected safety system can be satisfied by leveraging MEC technology. Their simulations are based on some VRU signaling, link, and latency models that they propose in their paper. Qin et al. [29] develop a collaborative intrusion detection system (IDS) and reduce the computational burden of performing IDS on vehicles. Le et al. [19] discuss latency optimization based on three-tier offloading model of a communication network where a vehicle can offload computational tasks to MEC as well as cloud. To the best of our knowledge, none of these works have implemented and evaluated an E2E vision-based connected safety applications via a real-world 5G cellular communication and a MEC computation setup.

Video Streaming and Analytics: Significant work has been expended to improve the efficiency of video analytics pipelines [11], [30], [31], [32], [33], [34], [35], [36], [37], [38]. Edge/Cloud based video analytics seeks to run accurate but compute-intensive DNN inference on massive video feeds from cheap cameras. Liu et al. [11] propose a low latency object tracking system for AR devices based on compute offload over WiFi to a PC based edge cloud instantiation. Jian et al. [39] propose an edge computing based technique to reliably extract motion from video frames and use the motion to speed up video analytics. Works such as AdaFrame [40] and Reducto [38] propose frame filtering techniques to reduce the number of frames to be processed to improve the efficiency of analytics. One popular design paradigm of such protocols is to leverage the server-side DNN to extract useful feedback from video stream and use the feedback to inform how the camera should encode and stream the video in the future. However, such feedback channels can take extra round-trip time specially in wireless environments and may incur latency on the order of seconds [38], whereas we are interested in latency of 150 ms or less. Moreover, our goal in this paper is to implement a practical and lightweight platform adequate for implementing and evaluating 5G MEC-based performance of connected safety applications. Employing strategies similar to aforementioned works may help us reduce latency further but we leave that exploration as part of our future work.

Camera Sensing in Vehicular Networks: Several works have explored single-camera and inter-camera multi-object detection and tracking such as [3], [4], [41], [42], [43], [44]. Various versions of You Only Look Once (YOLO) algorithm such as YOLOv4 [25] and YOLOv8 [43] can be leveraged for object detection in video frames. Algorithms such as DeepSORT [45] and ByteTrack [46] can be used to track objects detected object detection algorithms like YOLO. To improve tracking of objects across video frames, works such as Tang et al.'s [42] introduce an adaptive appearance model to learn long-term history of visual features for each vehicle target. In a later paper, the authors also introduce CityFlow [41], where they fuse mobility correlation and vision based techniques to improve the accuracy of vehicle identification across different camera observations. Tong et al. [3] propose a similar system that leverages multiple deployed traffic cameras as a sensing network to trace vehicle movements and tries to reconstruct their complete trajectories. Note that our goal is not to compete with these systems and algorithms, but to develop a robust application-specific mono-camera based detection, localization, and tracking solution that is efficient and can reliably enable low latency connected safety applications requiring lane level localization accuracy and can be used to evaluate their performance on 5G MEC-based system setup.

III. Raven Overview

Fig. 2 shows Raven's high-level system architecture. The *infrastructure module* is responsible for acquiring video data from the camera and transmitting it to the edge server. The *edge module* is responsible for processing any incoming video data from the infrastructure to detect and track objects by leveraging the computational resources of the edge server. This module also generates messages containing information about the tracked objects that get transmitted over a 5G cellular link to the subscribed vehicles. Although not used in this work, the edge module is also capable of providing information back to the infrastructure (i.e., if infrastructure requests edge compute information) via the cellular link. The *vehicle module* is responsible for extracting information about the detected objects from any received JSON messages and put them into standard Basic Safety Messages (BSM) [23]. These BSMs are then used by an after market On-Board Unit (OBU) for threat assessment and generating potential safety warnings. Although out of this paper's scope, *Raven*'s *infrastructure module* is also capable of leveraging local compute to process images for object detection, tracking, and generation of BSM messages, which provides the flexibility to test P2P direct communication-based I2V implementations as needed.

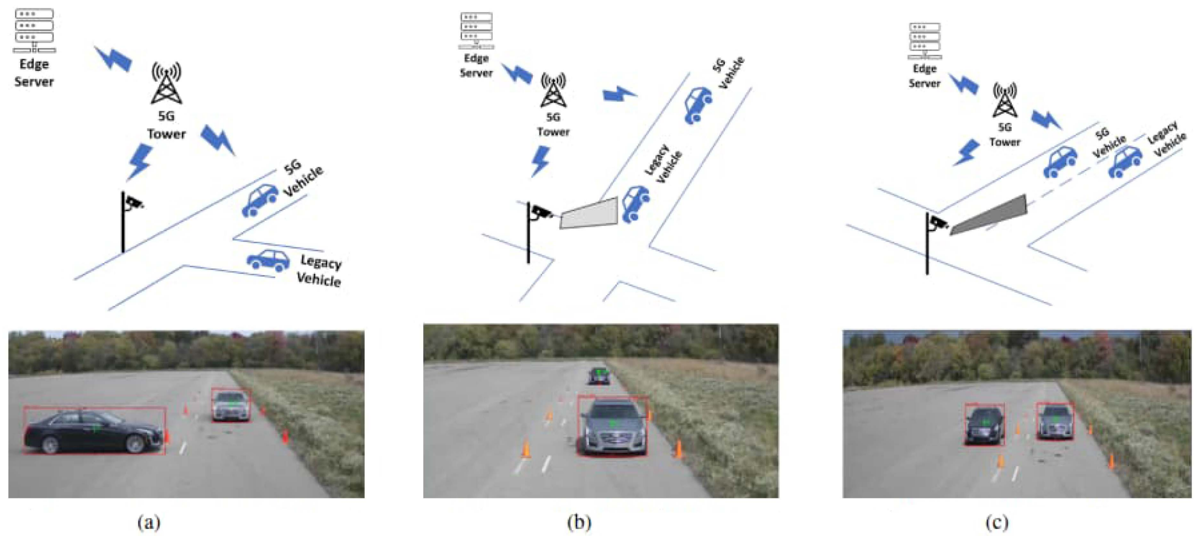


Fig. 1. Connected safety applications tested using *Raven* platform. (a) Intersection T-bone crash warning. (b) Stopped/disabled vehicle alert. (c) Vehicle-in-blind-spot warning.

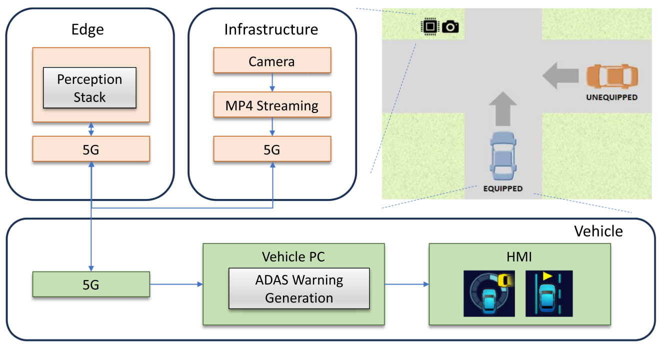


Fig. 2. High-level system architecture.

Fig. 3 shows a more detailed system diagram, which will be elaborated in the following sub-sections.

A. *Video Acquisition, Transmission, and Reception:*

Raven's video streaming pipeline is based on GStreamer [47] which is an open-source C++ based multimedia framework with extensible, modular, light-weight, and multi-threaded graph-based stream processing APIs. In particular, GStreamer's modular nature provides an intuitive way to link different stream processing "plugins" and arrange them in a "pipeline". Moreover, it allows us to leverage hardware acceleration using specialized plugins (e.g. hardware specific encoding and decoding libraries). *Raven* uses Real-Time Streaming Protocol (RTSP) [48] over UDP for video transmission.

Timestamping: Timestamps are required for two key purposes: (1) for the safety applications to accurately determine the position of a detected object for threat assessment, and (2) for E2E latency measurement between a sender (i.e. the camera) and receiver (i.e. any vehicle receiving safety messages from edge server). To achieve this, we extend GStreamer pipeline modules and RTP packet headers [49] such that the capture timestamps and IDs of the video frames pass through all the way from their moment of capture at the camera to their reception at the server. Although not implemented, we could adjust the percentage of RTP packets of a frame are tagged with timestamp and ID based on network conditions to reduce network load. This can be achieved via a feedback mechanism between the infrastructure and edge.

B. *Image Super-Resolution (SR) At Edge:*

To reduce bandwidth and latency, Raven provides an option to reduce video resolution before transmission on the infrastructure side followed by image resolution enhancement after reception at the edge server. However, due to real-time nature of ADAS safety applications, we need a fast SR algorithm. GPU resource consumption at the edge servers is another concern as such resources tend to be expensive. Although in our current implementation we test *Raven* using a single camera, yet one can easily imagine hundreds of cameras connected to a single edge server with limited GPU resources. Therefore, the SR method must have a low GPU footprint.

SR Algorithm: Standard resolution enhancing algorithms (e.g. such as linear or bicubic), apply a low–pass filter on a high resolution image created by inserting zeros between adjacent pixels in the low resolution image. Instead, machine learning based methods examples of high–resolution images to learn a mapping from low–resolution. Although there have been several advancements in deep learning based SR, most networks use hundred of thousands to millions of parameters. In this work, we utilize a minimal template matching (TM) self–attention based SR technique called eSR-TR introduced by Michelini et al. [24] that can perform image SR even with a single convolutional layer. The eSR–TM technique shows good performance at intermediate speed and image quality. In our current implementation of *Raven*, we use 3x upscaling factor for SR. Accordingly, we choose scale (s) = 3, templates (t) = 7, and filters (K) = 5 for configuring eSR-TM which leads to 4410 model parameters.

We customize the aforementioned SR model to be application specific, i.e., we train it to be good at reconstructing high resolution images from low resolution images of scenes containing

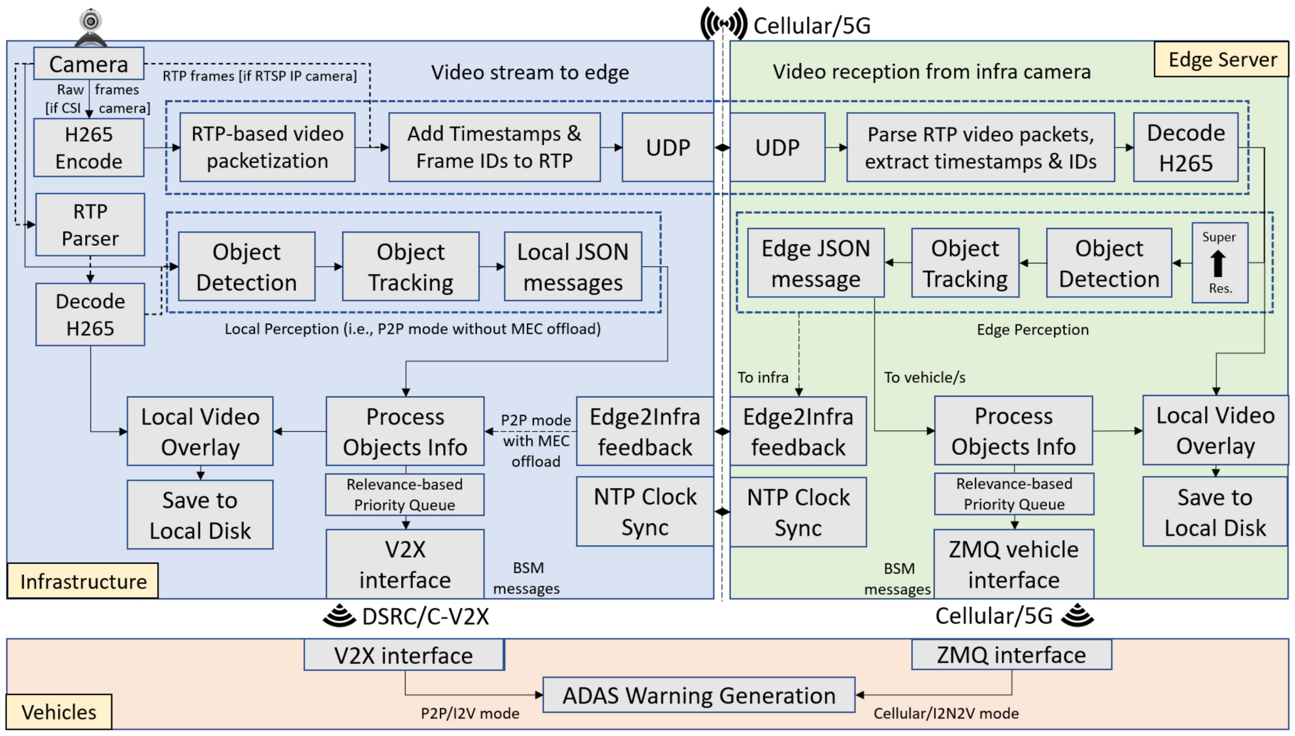


Fig. 3. Raven's system diagram showing both I2N2V and I2V (out of this paper's scope) capabilities.

typical roadside objects such as cars, trucks, buses, cyclists, and pedestrians. We achieve this by first curating a dataset consisting of more than 15 k images containing various types of roadside objects and then fine tuning the SR model by re-training it using the curated dataset.

C. *Object Detection, Tracking, and Localization:*

1) Object Detection in Camera Coordinates: Object Detection: Raven detects and locates objects in camera coordinates using the YOLOv4 algorithm [25]. The output of YOLOv4 is a vector of bounding boxes for each detected object in the image. We use YOLOv4 with (512 x 512) input size which is a good choice for detecting objects such as cars at distances of up to 100 m based on our experiments. For each detected object, the network gives 4 coordinates i.e. t_x, t_y, t_w, t_h and a confidence value between 0 and 1. These bounding boxes are then processed by our tracking algorithm that assigns IDs to the detected objects in a spatially and temporally consistent manner. Finally, the bottom middle pixels of the tracked bounding boxes are passed to the localization and state tracking module.

Dealing with Spurious Object Detections: One issue we encounter with YOLO is spurious detections of the same object either (a) in the same image and/or (b) in subsequent images but with different class labels instead. For example, sometimes a car can also be detected as a truck in the same image and we get two bounding boxes for the same object. Similarly, an object detected as a car in one image can be detected as a truck in the next frame. These issues can lead to generation of spurious safety messages and/or loosing tracking ID of an object - which are undesirable situations.

To deal with the first issue, we propose an efficient *neighborhood-based non-max suppression* (NB-NMS) algorithm on top of YOLO's standard NMS. The standard NMS algorithm runs on class specific bounding boxes only, i.e., it does not take into account that same object can be detected as multiple different objects, e.g., a car detected as both car and a truck with slightly different bounding boxes which leads to spurious detections. Our NB-NMS algorithm essentially performs NMS on all bounding boxes centered within a threshold pixel radius of each other independent of their class labels such that the computational complexity is constrained within that radius. If two objects seem to overlap significantly, we only keep the highest confidence detection.

As for each object in the scene there can be several spurious detections, we implement our NB-NMS algorithm using a KD-tree which is an efficient data structure to search for neighbors of an object. First, the algorithm inserts center pixel coordinates of all the detected objects in an image into a K-D tree. Second, it goes through all detections one by one and performs a radius search to obtain its neighbors in the K-D tree. Third, among all the neighbours of a detection (including itself), the algorithm determines maximum *intersection over union* (IOU) or overlap $IOU^i_{\max}$ of each i-th detection with any other detection in the neighborhood. Note that this is an $\mathcal{O}(N^2)$ operation, which is why we constrain the KD-tree radius search is within a few pixels (e.g. 15 pixels). Finally, the algorithm only keeps the detections (1) which has the highest confidence score among all the neighbors and (2) which do not overlap significantly with any of their neighbors (i.e. $IOU^i_{\max}$ is less than a given max IOU threshold e.g. 0.75 or 75%).

2) Object Tracking in Camera Coordinates: Our goal is to design a tracker that is reasonably accurate and robust in terms of detecting if two objects in two (or more in case of a history of image frames) different images are the same or not, and if they are the same, assign a consistent tracking ID. At the same time, we have to be mindful of the available compute resources at the MEC server and the latency sensitive nature of the safety applications, so the tracker needs to be computationally efficient and simple.

Initial ID Assignment for New Images: Objects detected in every new image do not have any ID at the beginning as YOLO is only capable of detecting objects and not tracking them. Therefore, a unique ID is assigned (from within a range e.g., 1-512) to each detected object in every new image. This initial ID assignment is based on a count that is incremented for each detection and overflows after reaching its maximum range. Therefore, the same object in a new and previous image/s will initially have two different IDs.

Tracking Algorithm: Our algorithm assigns object IDs in a spatially and temporally consistent manner based by maintaining a history of detected objects in the video frames. The current implemenation of *Raven* uses frame history of $F = 15$ frames, where detections corresponding to older frames are automatically removed from the queue. Let $\mathcal{B}_f$ denote a vector of all bounding boxes corresponding to detected objects in a video frame. Let $\mathcal{H}_f$ denote the current history of bounding box vectors corresponding to previous F video frames. For each new frame f, our tracking algorithm goes through every detection in $\mathcal{B}_f$ and checks if it matches spatially with any detection/s in history based on maximum distance threshold $D_{\max}^{track}$ and an minimum IoU overlap threshold O_{min}^{track}. Each detected bounding box in every bounding box vector in the frame history queue is searched, which is crucial to achieve *spatial consistency*. We make this search fast using a KD-tree such that only bounding boxes centered close to each other across frames are compared.

A new match is *found* if the center to center distance $d_{i,new}$ of the i-th detection in the current frame compared to another object in history is less than $D_{\max}^{track}$ (e.g., 75 pixels) and the overlap or IoU is within $O_{\max}^{track}$ (e.g., 0.25). We also compare the classes of the objects when trying to find a match. However, we only use class comparison if the overlap of those objects is lower than a threshold (e.g., 0.75). This is to address the issue where YOLO classifies the same object with different class labels across different video frames (e.g. Car or Truck). This can happen even after applying the aforementioned NB-NMS algorithm as NB-NMS algorithm does not work across images. Therefore, if the defections are close to each other and overlap significantly our tracking algorithm considers them the same object independent of the class label. Current best match is *updated* if $d_{i,new}$ is less than the previous update $d_{i,old}$. Note that if we just keep the candidate with highest overlap after the current best match is updated, we increase the risk of obtaining a wrong ID as we are searching for matches across a history of image frames and some detections in those frames may undergo temporary occlusions or the corresponding images may experience significant packet losses. Therefore, there is a need to maintain a map of candidate matches for each detection from which we can select the best candidate in a *temporally consistent* manner.

To achieve temporal consistency, our algorithm allows each detection to have more than one possible matches in the frame history while meeting the proximity, overlap thresholds, and class label requirements mentioned before. All the candidate matches are then maintained in a *candidate indices map* $\mathcal{C}_f$ which is a list of maps corresponding to the detections (or bounding boxes) in $\mathcal{B}_f$ such that each detection in $\mathcal{B}_f$ has a unique associated entry in $\mathcal{C}_f$. For each map in the list, the *keys* are the matched candidate IDs and *values* are their count (i.e., number of times a detection matches closely with an object with same ID in $\mathcal{H}_f$). The map corresponding to each detection is searched for ID with the highest count, which becomes the most *temporally consistent* tracking ID of that detection in the current frame.

Resolving Duplicates: As mentioned before, our tracker tries to assign object IDs in a spatially and temporally consistent manner based on video frame history. Towards that goal, the tracking algorithm checks for each detected object in a new image frame if that object matches with any other detected objects in the frame history $\mathcal{H}_f$. The matching between candidates is performed based on three key metrics: (1) the center-to-center distance, (2) amount of overlap, and (3) class ID of the bounding boxes corresponding to those objects. However, as roadside objects can be very close to each other in real-world crowded scenarios and our matching thresholds are not perfect (i.e., based on heuristics), multiple objects in the new image frame may match with the same object in $\mathcal{H}_f$ and therefore can get the same ID. Moreover, as each detected object in a new frame is initially assigned a random ID from a range (e.g., 1-512), it is entirely possible that one of those newly assigned IDs is already in the $\mathcal{H}_f$.

To eliminate such duplication issues, after the main steps of spatial-temporal matching process are completed for the new image frame, the algorithm first creates a map where the keys are all the newly assigned IDs for objects in the current frame and values are vector of match scores (i.e., a Boolean variable signifying if a match was found or not and the highest overlap or IoU score). Finally, the algorithm goes through each ID and resolves duplicates as follows: (1) If an ID is candidate for exactly one object in the current frame, that object's ID goes unchanged. (2) If an ID is candidate for multiple objects, it is assigned it to the object with the highest overlap score. (3) The remaining objects are then one-by-one (incrementally) assigned IDs that are taken randomly (without replacement) from the range (e.g., 1-512) until those IDs don't conflict with any other entries already in the map and the map is updated accordingly (i.e., object info is removed from the old ID key in the map and added with the new ID key). The algorithm stops once all the duplicates are resolved.

Discarding Old Detections: As the algorithm processes live stream, it is possible that some frames in the stream are delayed significantly due to network delays. Therefore, detections corresponding to any frames $\geq t_{old}^{track} = 1000\,\text{ms}$ away from the latest frame are also removed. Note that when detections corresponding to a certain frame in history are removed from the queue, their corresponding entries in the KD-tree (used when searching for ID matches) are also removed.

3) Coordinate Conversion: To compute the real world location of objects detected in an image, we have to convert the image coordinates of the object to world coordinates. To do this, we compute a transformation matrix (known as Homography) between the world frame (we assume we are monitoring planar areas of a road in GNSS coordinates) and the camera frame. This is a one time calibration effort. To do this calibration, we need calibration point pairs, i.e., markers in image frame and their

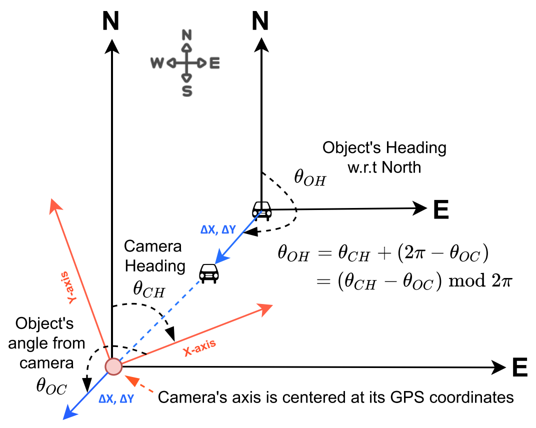

N
N
N
W
E
S
Object's Heading w.r.t North
θ_{OH}
E
ΔX, ΔY
Camera Heading
θ_{CH}
$\theta_{OH} = \theta_{CH} + (2\pi - \theta_{OC})$
$= (\theta_{CH} - \theta_{OC}) \bmod 2\pi$
Y-axis
X-axis
Object's angle from camera
θ_{OC}
E
ΔX, ΔY
Camera's axis is centered at its GPS coordinates

Fig. 4. Translating vehicle heading to global coordinates.

corresponding one-to-one matched locations in the real world (e.g., GNSS Lat/Lon coordinates). This conversion assumes linearity in GNSS coordinates across the patch of the road area being monitored. Moreover, as we assume that the monitored area is flat, all the points used for calibration are coplanar. However, the points must not be collinear to avoid trivial solutions. We collect 10 point pairs from diverse zones of the monitored area e.g., we choose points from far, mid, and close zones, and leverage OpenCV library to compute the Homography. To improve the robustness of the estimate, we use three different methods i.e., (1) a regular method using all the points, (2) Least-Median robust method, and (3) RANSAC-based robust method and select the output of the one that gives minimum reprojection error.

4) Localization and State Tracking in World Coordinates: To address the non-linearity involved in converting points from image to world-coordinate frame, we implement an Uncented Kalman Filter (UKF) [50] based algorithm to track motion states (position, speed, acceleration, and heading) of the objects detected in the video frames. We use an X/Y coordinate system centered at the camera (i.e., the reference point, see Fig. 4) for this. The filtered position is later converted to GPS/GNSS coordinates before sharing with the vehicles. We implement both constant velocity and constant acceleration models. Note that although the state-transition function is linear for both of these models, yet the camera observation is nonlinear. This non-linearity is introduced when camera pixels are converted to X/Y world coordinates, which is why we choose UKF. We keep a map of UKF filters corresponding to all the tracked objects where the keys are tracking IDs of those objects. UKF map items are removed if an object is not seen for more than $T_{rem}^{ukf} = 5$ seconds. The output is not shared with subsequent modules unless state tracking has observed the object at least $C_{obs}^{ukf} = 10$ times in last $T_{obs}^{ukf} = 3$ seconds. The values of these thresholds can be tuned according to the video frame rate (i.e., FPS). Next, we give some details around our implementation of UKF.

Just like in regular KF, UKF is also composed of two steps: *predict* and *update*. Let $\mathbf{F}$ be the state transition matrix, $\mathbf{Q}$ be the process noise matrix (following [51]), $\mathbf{h}$ be the measurement function, and $\mathbf{R}$ be the measurement noise matrix. In the predict step, prior is computed using the process model $\mathbf{F}$. To do this, first a set of *sigma points* $(\mathcal{X}_i)$ and their corresponding weights $(\mathcal{W}_i)$ are computed using previous state $\mathbf{x}_{k-1}$ and covariance matrix $\mathbf{P}_{x,k-1}$ via a sigma point generation function g. UKF uses sigma points to sample from a given Gaussian distribution defined by a mean and a covariance matrix. We use the method proposed by Rudolph Van der Merwe in [50] that generates $2n+1$ sigma points X_i around previous state $\mathbf{x}_{k-1}$, where $n =$ dimension of the state space. $n = 4$ and $n = 6$ in the case of constant velocity and constant acceleration models, respectively, that correspond to position, velocity, and/or acceleration states in 2D. Moreover, we choose $\alpha = 0.75$, $\beta = 2$, and $\kappa = 3 - n$. More details of these constants can be found in [50]. These sigma points are then transformed through the process model. Finally, the state mean $\overline{\mathbf{x}}$ and covariance $\overline{\mathbf{P}}_x$ of the prior is computed using the unscented transform (UT) [50] on the process transformed sigma points. (1) and 2 summarize the aforementioned sigma point generation and prior distribution estimation steps:

$$(\mathcal{X}_i, \mathcal{W}_i) = g(\mathbf{x}_{k-1}, \mathbf{P}_{x,k-1}) \tag{1}$$

$$(\overline{\mathbf{x}}_k, \overline{\mathbf{P}}_{x,k}) = UT(\mathbf{F}\mathcal{X}_i, \mathbf{Q}) \tag{2}$$

After predicting the state mean and covariance, the next step is to predict measurement mean and covariance. To achieve this, first the sigma points of the prior (i.e., transformed by process model $\mathbf{F}$) are converted into measurement sigma points using the measurement function. We choose to keep the measurement space in camera pixel coordinates. Therefore, the measurement function $\mathbf{h}$ is an inverse conversion taking the sigma points in X/Y coordinates to GNSS coordinates and then applying inverse camera Homography giving pixel values in camera frame. Similarly, the measurement noise $\mathbf{R}$ is also provided in pixels (noise along the camera x-y pixel coordinates) that we estimate experimentally by parking a vehicle at a few locations and running YOLO on all the logged images. The mean $\overline{\mathbf{z}}$ and covariance $\overline{\mathbf{P}}_z$ of these measurement sigma points are then obtained after applying the UT:

$$(\overline{\mathbf{z}}_k, \overline{\mathbf{P}}_{z,k}) = UT(\mathbf{h}(\mathcal{X}_i), \mathbf{R}) \tag{3}$$

Additionally, the cross covariance of the state and the measurements is computed using the measurement sigma point means and covariances obtained in the previous steps:

$$\mathbf{P}_{xz} = \sum \mathcal{W}_i\{\mathbf{F}\mathcal{X}_i - \overline{\mathbf{x}}_k\}\{\mathbf{h}(\mathcal{X}_i) - \overline{\mathbf{z}}_k\}^T \tag{4}$$

In the update step, first the Kalman gain is computed using the cross and measurement covariances as $\mathbf{K}_k = \mathbf{P}_{xz}\overline{\mathbf{P}}_{z,k}^{-1}$. Finally, new state $\mathbf{x}_k$ and covariance $\mathbf{P}_k$ are updated as follows:

$$\mathbf{x}_k = \overline{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\overline{\mathbf{x}}_k)) \tag{5}$$

$$\mathbf{P}_k = \overline{\mathbf{P}}_k - \mathbf{K}_k\overline{\mathbf{P}}_{z,k}\mathbf{K}_k^T \tag{6}$$

One issue with using a KF is the overshoot in velocity and/or acceleration estimation when an object is detected the first time. The problem is specially exacerbated for objects that are further away from the camera as velocity uncertainty gets higher. Such overshoots, even if they happen for a short duration, lead onboard threat assessment algorithms of receiving vehicles to generate a premature warning. To avoid this issue, we leverage standard equations of motion only to estimate the velocity and acceleration of an object in the initial few consecutive frames (e.g., 10 frames) it is detected in. The first half of those frames (e.g., 5 frames) are used to estimate velocity only and the second half are used to estimate both velocity and acceleration. These values of acceleration and velocity are then used to initialize the UKF.

Heading Estimation: Heading of the vehicle is derived with respect to the reference point using change in motion along X and Y direction as shown in Fig. 4 using $\tan^{-1}(\frac{\Delta y}{\Delta x})$. The Δx and Δy are computed using consecutive position estimates from UKF output. Object's heading is updated if a significant is detected in its position. We apply a median filter (window size = 5) to smooth out abrupt changes in heading estimation.

D. Safety Messaging Interface:

Raven obtains the original capture timestamp, tracking ID, position (in GNSS coordinates), speed (in kph), and heading (with respect to the North as shown in Fig. 4) values of all detected objects in the video frame from the state tracking module. We assume that the monitored area is approximately flat so the elevation value (in meters) can be predetermined and stays constant. *Raven*'s safety messaging module packages this information into JSON format and shares it with the vehicles that have subscribed to *Raven*'s service. The receiving vehicles extract this information and put them into the standard Basic Safety Message (BSM) format [23] corresponding to each detected object. These BSMs are forwarded to the vehicles' BSM message processing module which uses that information for threat assessment and generating any warnings.

Our current implementation uses ZeroMQ (ZMQ) dynamic discovery mechanism that allows publishers and subscribers to discover each other on the network without requiring any configuration or centralized service. This mechanism works by using a proxy, which acts as a message router and mediator between publishers and subscribers. The proxy uses the publish-subscribe (PUB-SUB) pattern to distribute messages from publishers to all interested subscribers. However, instead of requiring subscribers to explicitly connect to publishers, the proxy dynamically discovers publishers and subscribers as they come online and offline. To achieve this, the proxy uses two special sockets: a frontend socket and a backend socket. Publishers connect to the frontend socket and send messages to the proxy, while subscribers connect to the backend socket and receive messages from the proxy. The proxy then dynamically routes messages between the frontend and backend sockets based on the topic of each message. In reality, we need to limit the number of subscribers and publishers on a single proxy and will need to have multiple such proxies to scale the system, as putting too much pressure on a single proxy will lead to increased buffering and latency. Moreover, publishers and subscribers can be allocated based on their geographical location (e.g., based on proximity to an intersection). The approximate locations of the receiving vehicles can be used by the server to automatically subscribe vehicles to safety messages from their relevant publisher. However, the implementation of these ideas is out of the scope of this work.

Relevance-based Priority Queuing: We propose a *relevance priority* based message queuing strategy where the information about the most relevant objects is transmitted first from MEC server to vehicles through ZMQ interface. This increases the chances of higher priority messages delivered in time before they are dropped as MEC messages remain in queue only for a small amount of time and then dropped as they become obsolete quickly. In general, priority can be defined based on regions of interest (ROI), distance, speed, heading, size, and type of the detected object. However, in our current implementation, we define a simple priority metric:

$$p_r = \frac{v}{d} \cdot \cos(\theta) \cdot e^{(\alpha|v-\mu_v|)} \tag{7}$$

Here v is vehicle speed in kph, d is distance in meters from point of interest (e.g., an intersection), θ is the heading (e.g., with respect to intersection), μ_v is normal speed (e.g., speed limit) of the monitored road, and α is some predetermined constant. For objects that are equally close, this function prioritizes higher speed objects. For objects with similar speed, it prioritizes objects that are closer (e.g., to an intersection). The exponential term ensures that objects whose speed deviates significantly with respect to the normal speed on the road (e.g., a stopped car) get a higher priority. Finally, we employ a *zero buffering* strategy such that the messages corresponding to a new image frame are immediately inserted in the queue after clearing the queue of any old messages.

TABLE I
HARDWARE SPECIFICATIONS

Location	Hardware	Specifications
Infrastructure	CSI Camera	MIPI-CSI-2, 4k, 30fps
Infrastructure	IP Camera	Hikvision, 1080p, 30fps
Infrastructure	PC	Intel Xeon 3.1GHz, 32GB
Edge	Server	Intel 32 cores, 64GB, Nvidia Tesla V100
Vehicle	PC	Intel NUC 2.9GHz, 16GB
Vehicle	5G	Samsung Note 10 5G (as hotspot)
5G Tower	5G Equipment	4G LTE(1900MHz), 5G mmWave (39GHz) and 5G Sub6(850MHz) uplink 6Mbps, downlink 35Mbps

IV. IMPLEMENTATION

We implement *Raven* on commodity off-the-shelf hardware. The hardware list and specifications are summarized in Table I. The overall implementation consists of more than 6500 lines of code on top of existing open-source libraries that support *Raven*. The edge service is deployed as a Docker container to allow scaling up the processing infrastructure using Docker Swarm or Kubernetes if needed.

Infrastructure Hardware: In terms of camera compatibility, our system currently provides two options: (1) MIPI-CSI-2 camera and (2) IP camera. From the CSI camera, we obtain raw frames that we encode and packetize into RTP packets (Fig. 3). For the IP camera, we configure its video streaming settings to use RTP format, so that the packets coming from the camera are directly forwarded to the edge server (Fig. 3). We choose MIPI-CSI-2 camera for this work mainly due to its flexibility and resolution options. We connect the CSI camera to an NVIDIA Jetson Nano that we use to implement our video transmission pipeline, as mentioned in Section III-A.

Private 5G and MEC Testbed: Since public commercial 5G and MEC services are not widely available, we have installed and created a private 5G and MEC testbed with the help of a commercial service provider to study and validate 5G/MEC automotive applications. As shown in Fig. 5, the testbed consists

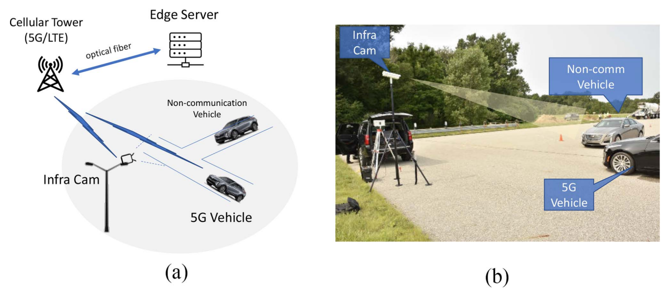


Fig. 5. Cellular and MEC based Setup (I2N2V).

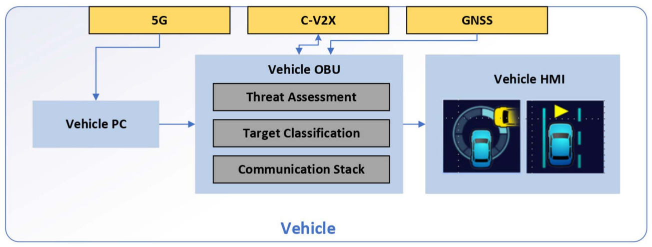


Fig. 6. Experimental Vehicles Capabilities

of a 5G tower, an edge server, and experimental vehicles. The 5G tower operates in three modes: 4G LTE (1900 MHz), 5G mmWave (39 GHz) and 5G Sub-6 (850 MHz). The edge server is linked to the 5G tower via optical fiber cables. On the experimental vehicle, a Samsung 5G phone is used for communication with the tower. The phone is USB-tethered to vehicle on-board computers. The 5G tower and the vehicle test area are around 700 m away. The maximum uplink bandwidth experienced is around 6 Mbps. We leverage an NVIDIA Tesla V100 GPU at the server to run the most accurate version of YOLOv4 model with average object detection latency of 21 ms, which is more than enough to support a 30 FPS video stream.

Safety Application Experimental Vehicle: For this study we use two standard production vehicles. The vehicle capable of receiving detected objects information from the MEC server is called the *host vehicle* (HV) and the vehicle that poses a possible threat to the HV is called the Remote Vehicle (RV). Both vehicles are retrofitted an after-market On-Board Unit (OBU), but the HV is retrofitted with an additional Vehicle PC. HV's Vehicle PC is responsible for receiving information about the neighboring objects (e.g. the RV) tracked by the MEC server and then forwarding that information to the HV's OBU. The HV's OBU performs target classification on the received tracked objects. It then evaluates potential threat of collision from/with the neighboring objects and issues a warning - a combination of visual, audio and haptic to the driver. The HV's OBU also logs its own location (based on its on-board GNSS module) as well as the location of the neighboring objects (e.g. the RV, based on the information coming from the MEC server). Fig. 6 shows the capabilities of our experimental HV. RV's capabilities are similar to HV except for the 5G communication capability and Vehicle PC. The RV's OBU also logs its own location for ground truthing purposes as discussed in Section V-A. Although both experimental vehicles are capable of C-V2X direct communication, yet we only use the cellular communication capability for this paper.

V. Evaluation

We evaluate *Raven* in terms localization and state tracking accuracy, end-to-end (E2E) latency, and timely warning generation for the safety applications. We define E2E latency as the latency experienced from the time an image is captured to the time a vehicle receives messages about the road objects detected on the edge server. We evaluate the tracking accuracy and latency for both SR and non-SR based approaches. The image resolution for SR experiments is 640×480 and for non-SR experiments 1920×1080. We use variable bitrate encoding for both SR and non-SR experiments capped at 2 Mbps.

A. Localization and State Tracking:

Grouth Truthing: The RV's OBU provides periodic information about its dynamics that we use as ground truth information. The position and heading information comes from an inbuilt GNSS receiver. Meanwhile the speed information comes from the vehicle CAN bus. The OBU combines the information from the two sources and generates a log file synchronized to UTC time. We discuss the accuracy of *Raven*'s localization and state tracking, i.e., distance, speed, and heading, when compared to the RV's OBU ground truth logs.

1) Distance Accuracy: Fig. 7(a) shows overall distance accuracy. The distance is measured with reference to the GNSS coordinates of the infrastructure camera installation. We observe that the variance of distance error is higher in for the SR experiments. To see details, let us look at the lateral and longitudinal error in Fig. 8(a) and (b) across different range bins from the reference point. Lateral error stays similar across all range bins, but is higher for SR case. Longitudinal on the other hand is higher that lateral error for both SR and no-SR cases. This is because of limited depth perception with monocamera where objects at distance have higher depth uncertainty, even with a well calibrated system. Another observation in non-SR case is that the longitudinal error stays similar across range bins but for SR experiments it increases. This is because lower resolution increases longitudinal distance uncertainty even further. Note that distance accuracy is also affected by YOLO bounding box noise, i.e., for the same object in the same image YOLO gives slightly different bounding box. At longer distance, even slight variation in YOLO bounding box location can lead to significant changes in distance estimation. One more observation is that SR reduces the range of detection as no objects are detected beyond 65-75 range bin but non-SR can detect up to 85 m bin.

2) Speed Accuracy: Fig. 7(b) shows overall speed accuracy. We see that the variance of speed estimation is is higher in the SR case. Fig. 10(a) and (b) show speed accuracy per bin for non-SR and SR experiments respectively. We see a similar trend in speed accuracy, but this time both non-SR and SR show similar trends. The speed error in SR case is more significant compared to non-SR due to resolution reduction. Speed accuracy is impacted because objects at distance tend to have higher distance uncertainty so even small pixel movement in YOLO detection can significantly change the velocity estimate. In SR case, this impact is enhanced.

3) Heading Accuracy: Fig. 7(c) shows overall heading accuracy. For heading computation as well, the variance for SR

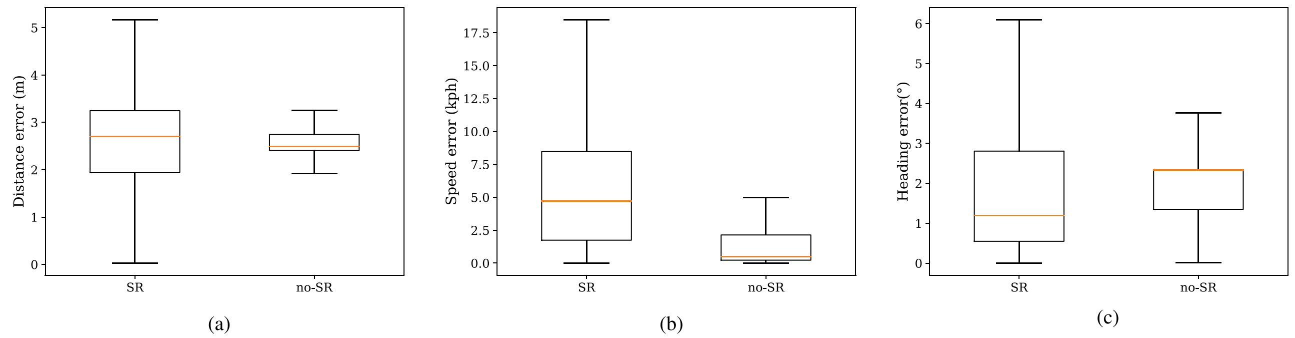


Fig. 7. Box plot statistics of distance, speed, and heading error across all the SR and no-SR experiments. (a) Overall distance error box plot. (b) Overall speed error box plot. (c) Overall heading error box plot.

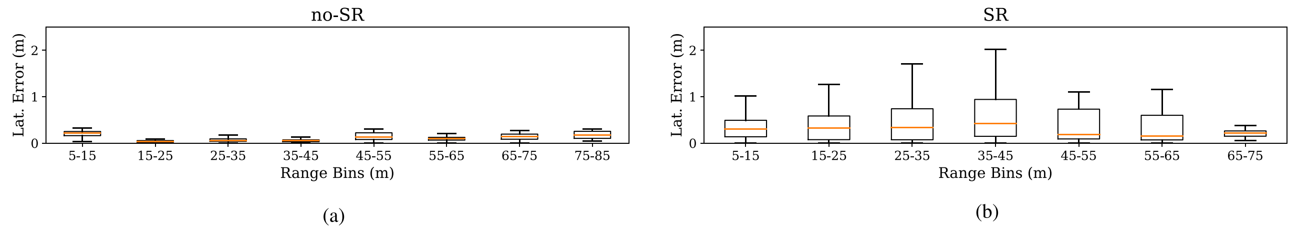


Fig. 8. Lateral distance error per range bin without and with super resolution. (a) Lateral error per range bin without super resolution. (b) Lateral error per range bin with super resolution.

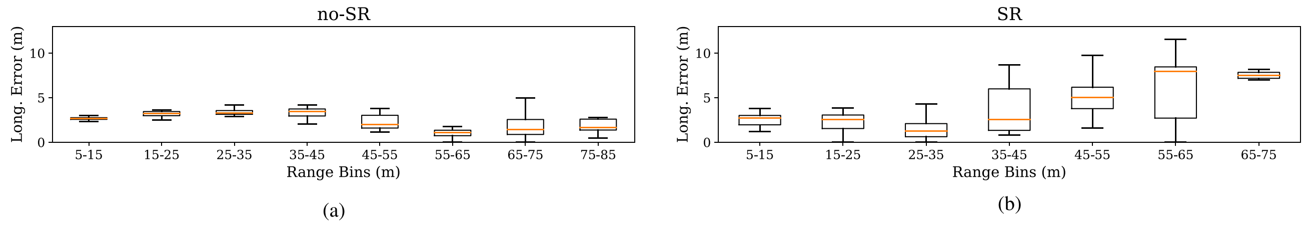


Fig. 9. Longitudinal distance error per range bin without and with super resolution. (a) Longitudinal error per range bin without super resolution. (b) Longitudinal error per range bin with super resolution.

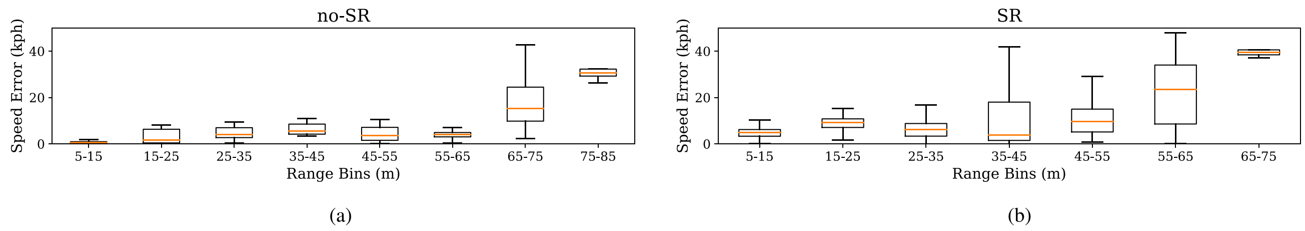


Fig. 10. Speed error per range bin without and with super resolution. (a) Speed error per range bin without super resolution. (b) Speed error per range bin with super resolution.

is higher but still within reasonable limits. Fig. 11(a) and (b) show heading accuracy per bin for non-SR and SR experiments respectively. We observe that heading accuracy stays consistent for non-SR case, increases for SR case, which can again be attributed to resolution reduction as well as YOLO detection noise. As mentioned in Section III-C4, heading is estimated using speed changes along lateral and longitudinal directions with respect to the reference point. Therefore, errors in speed estimation lead to heading errors, especially when the speed change errors are along the lateral direction.

In general, we observe that the position, speed, and heading accuracies are higher for the non-SR approach compared the SR approach. However, as we show later Section V-C, both the accuracy and range of detection of the SR based approach

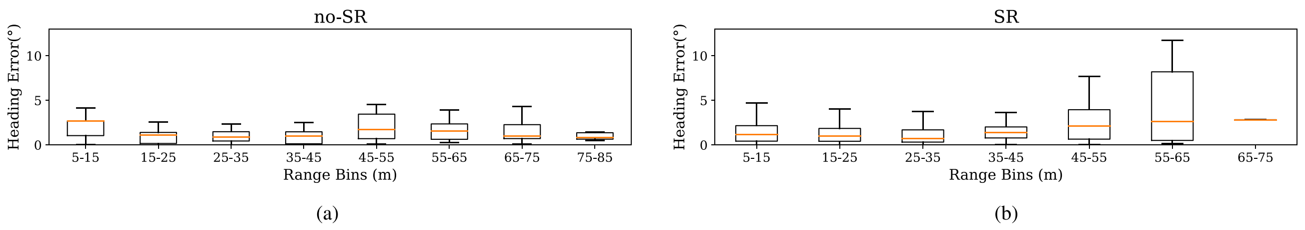


Fig. 11. Heading error per range bin without and with super resolution. (a) Heading error per range bin without super resolution. (b) Heading error per range bin with super resolution.

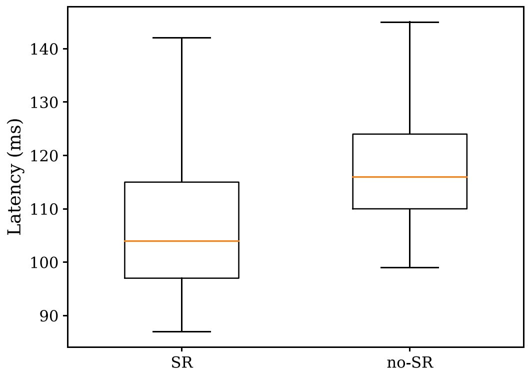


Fig. 12. Latency with and without Super Resolution.

is adequate for effective working of the implemented safety application. An added advantage of the SR approach is reduced E2E latency, which we discuss next.

B. Latency:

Fig. 12 shows the box plots of latency across all experiments for both SR and non-SR cases. We can see that the median latency is lower for the SR cases (103 ms) compared to non-SR cases (117 ms). Super resolution trades off using more edge compute resource for the SR model discussed in Section III-B with lower communication latency. Although sensing lower resolution frames reduces latency by $\approx$ 22 ms, yet the SR module introduces an extra $\approx$ 8 ms of compute latency. This leads to an overall latency reduction of $\approx$ 14 ms. Note that we show averaged results over all experiments done in the same settings but in a private 5G communication and MEC setup. However, in a public 5G and MEC setup, the latency will also depend on network conditions that are not in our control, e.g., congestion, which is out of this paper's scope.

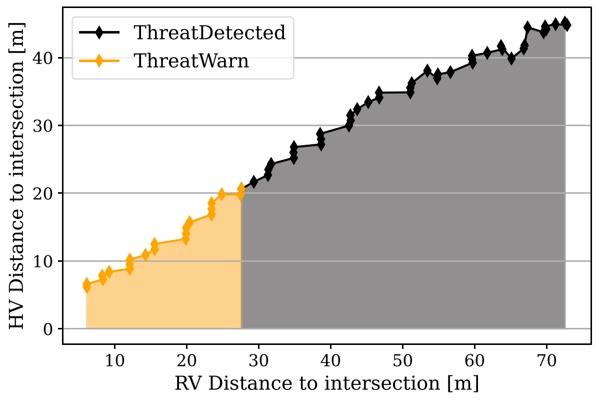
C. Performance of Vehicle Safety Applications:

Raven's goal is to provide detected objects information to the subscribed vehicles such that they can detect potential collision threats and warn the human or autonomous agent in a timely manner. During all the test runs of the implemented safety applications, the driver was alerted in a timely manner, indicating that the applications work effectively and the accuracy and precision of the sensing, localization and tracking pipeline is adequate. Note that the timing of warnings generated by these applications is based on parameters that are configurable and are usually adjusted based on driver clinics before production. As such a driver study is out of the scope of this work, we experimentally determine those parameters for testing and demonstration purposes. Moreover, we observe that both SR and non-SR based approaches adequately support all of the implemented safety applications. Therefore, for brevity, we present the application performance results for the SR based tests only. In the following experiments, the RV is assumed to have no communication capability compared to the HV. *Raven*'s performance is determined by its capability to track and share the RV's state with the HV.

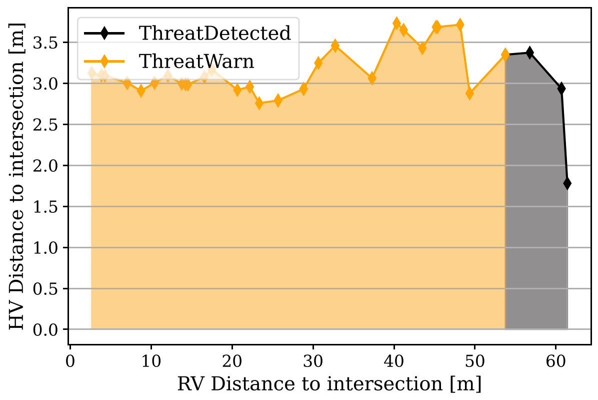
1) Intersection Movement Assist (IMA): The IMA safety application (Fig. 1(a)) warns the driver of an HV when it is not safe to enter an intersection due to a high crash probability with any intersecting RVs. This safety application would provide safety warnings that help drivers avoid intersection T-Bone crashes. We show results for both static and moving HV IMA scenarios, each with and without super resolution.

Moving HV IMA Scenario. Both vehicles approaching the intersection or the moving IMA is a scenario where the HV is approaching the intersection when visibility may be limited for example by presence of another vehicle; and RV is approaching the intersection from left or right of the HV. The HV driver will receive a warning from the IMA feature indicating that a conflict is predicted with the RV as it crosses the intersection. The timing of the warning is expected to be set such that the driver of the HV can avoid a collision with the approaching RV. This test scenario involves the HV approaching the intersection, when simultaneously the RV is approaching the intersection perpendicularly from left. Both HV and RV speeds are in the range of 30 MPH to 40 MPH. The results for IMA moving HV, are shown in Fig. 13(a). The black lines in the figure correspond to observation samples when the IMA application detects the RV as a potential future threat but not currently an imminent threat for collision. As the HV and RV further approaches the intersection, the threat becomes imminent and a warning is issued to the driver (orange lines), so that the HV driver can stop safely.

Static or Creeping HV IMA Scenario. Stopped/Creeping HV at intersection or static IMA is a scenario where the HV is stopped at an intersection and visibility may be limited, for example by other stopped vehicles. In such case when RV approaches the intersection, from left or right of the HV, the driver receives a warning from the IMA feature indicating that a conflict is predicted with RV as it crosses the intersection. The timing of the warning is a function of RV's speed and is set such that the driver of the HV can avoid a collision with the approaching RV. This test scenario involves the HV stopped and



(a)



(b)

Fig. 13. IMA with Super Resolution. (a) IMA Moving. (b) IMA Static.

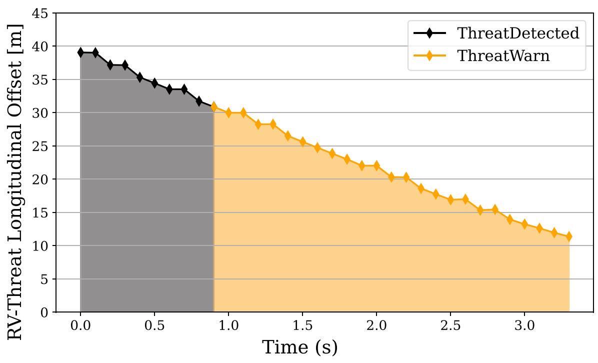
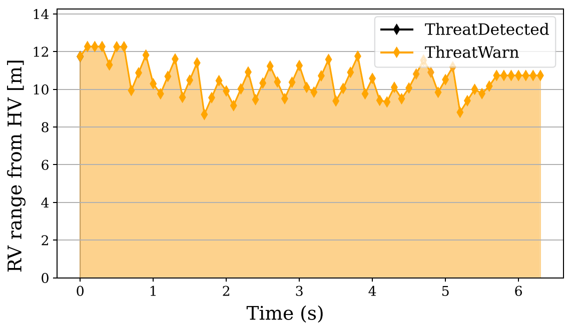
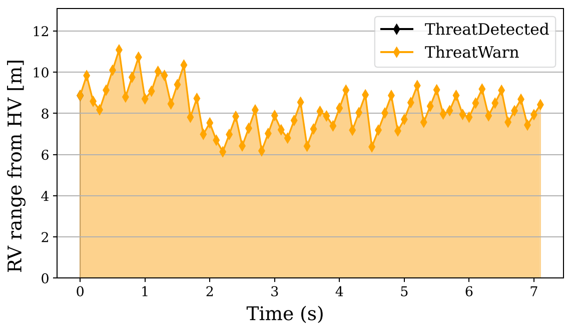


Fig. 14. FCW results with SR.



(a)



(b)

Fig. 15. BSW results with SR. (a) Test 1. (b) Test 2.

trying to creep into the intersection (for example coming out of a parking lot), when simultaneously the RV is approaching the intersection perpendicularly from left. The RV speed is in the range of 30 MPH to 40 MPH. The results for Static IMA are shown Fig. 13(b). The black lines in the figure correspond to observation samples when the IMA application detects the RV as a potential future threat but not currently an imminent threat for collision. As the RV further approaches the intersection, the threat becomes imminent and a warning is issued to the driver (orange lines).

2) *Forward Collision Warning (FCW):* The FCW application (Fig. 1(b)) warns an HV driver of an impending rear-end crash with an RV (e.g., a stopped or disabled vehicle) directly ahead in the same lane and direction of travel to help the driver avoid or mitigate such crashes. The FCW tests involve an HV approaching an RV, which is stopped in the same lane as the HV. The HV's speed is in the range of 30 MPH to 40 MPH. The HV receives a warning from the FCW safety application when there is imminent danger of a rear-end collision with the stopped RV in its lane of travel. The timing of the warning is set such that the driver of the HV can make a safe stop and avoid crashing with the RV. FCW test results are shown in Fig. 14.The black lines in the figure correspond to observation samples when the FCW application detects the stopped vehicle as a potential future threat but not an imminent threat for collision. As the HV further approaches the RV, the threat becomes imminent and a warning is issued to the driver (orange lines).

3) *Blind Side Warning (BSW):* The BSW safety application (Fig. 1(c)) warns the driver of HV during a lane change attempt if the blindspot zone into which the HV intends to move into is, or will soon be, occupied by an RV traveling in the same direction. When HV is not attempting a lane change, the BSW application still provides advisory information if an RV in an adjacent lane is positioned in a blindspot zone of the HV. During the BSW safety application testing, the HV and the RV were driven at 30 MPH to 40 MPH speed, with RV coming/staying in HV's left side blindspot. Note that this blindspot zone is configurable, and was set to be up to 15 m for this proof of concept implementation. The HV driver attempts to make a lane change by turning the left turn indicators on. The BSW warning received by the driver and relative distance between HV and RV are captured in Fig. 15. The HV had it's left turn indicator on throughout this period and due to this BSW application status in the figure stayed at "warning".

VI. Discussions and Future Work

Raven is an early step towards determining feasibility of 5G MEC-enabled connected vehicle safety applications. There is obviously room for continued research in various perspectives. In this section, we provide commentary on the limitations of our work and discuss avenues of future research.

Scalable public deployments and testing: The experiments reported in this paper were conducted in a private vehicle test facility with a private 5G tower and a private edge server. We did the experiments in this private setup for two reasons: (1) public safety, because during the tests, we make sudden maneuvers to test different crashing scenarios, such as, intersection T-bone crash, stopped vehicle crash, blind spot crash, etc.; and (2) public 5G and edge computing services are not widely available in the country. For example, our test facility is not covered by public 5G signals and commercial edge services. Consequently, our experiments may have two limitations: (1) the end-to-end latency in public 5G/edge environments and on public roads may be higher due to communication hops and network bandwidth; and (2) the scalability for communication and edge computing resources is an issue still to be investigated, as in a public 5G MEC deployment, there maybe several infrastructure cameras and a number of road users connected to the server. This can cause congestion in the network which in turn introduces latency. We leave scaling *Raven* to meet the needs of such real world scenarios and related evaluation as part of future work. We believe that Network Slicing [52], which is a key upcoming feature of 5G networks, will play a key role in guaranteeing the quality of service (QoS) by providing dedicated network resources, which can be dynamically customized to meet the bandwidth and latency requirements of specific connected safety applications.

Comparison with existing V2X standards: Our primary objective is to explore the feasibility of 5G MEC-based applications. We are in no way claiming superiority over setups based on existing standards such as DSRC or C-V2X. Our future research will involve developing V2X-based setups and comparisons with relevant public standards, with results to be shared in subsequent papers. It's important to note that deployment technology selection will be influenced by both availability and performance, factors and will be determined by market dynamics. Note that we have designed our system with the capability to use V2X instead of sending information to MEC, as depicted on the left hand side of Fig. 3. While we have the capability to utilize V2X technology, we have not conducted experiments in this regard yet, leaving it as a direction for future work.

Design of transmission priority metric: As mentioned in Section III-D, our current implementation employs a priority based message queuing strategy where the information about the most relevant objects is transmitted first from MEC server to vehicles. We acknowledge that in general, there can be several metrics that can be considered when defining such a priority e.g., based on interactions among multiple objects, regions of interest (ROI), distance, speed, heading, size, types of the detected objects, and so on. For example, one feasible way to obtain a measure of such interactions would to develop a machine learning algorithm that can detect dangerous interactions on the road based on which we can develop a better priority metric. However, in our current system, all the threat assessment logic resides on the receiving vehicles. The MEC server is merely a source of information for those vehicles. In this initial work, we have proposed a relatively simple model to define priority of road-side objects based on their speed, distance, and heading. Although we acknowledge that there is room to improve the priority metric, yet, we leave performing more involved threat assessment and prioritization at MEC server as part of future work.

System flexibility and modularity: We want to clarify that while our system is designed with a flexible modular framework, the primary goal of this work is not to create an open platform where external researchers and developers can contribute software or hardware modules. Instead, the focus of this work lies in investigating the feasibility of leveraging 5G cellular networks and mobile edge computing servers for automotive active safety applications and sharing our experiences and insights with the research community. That said, our platform does offer flexibility for future enhancements and adaptations. For instance, we could develop new algorithms or safety applications by upgrading or replacing modules within the system architecture (as depicted in Fig. 2 and Table I).

Accuracy of object tracking in images: In this work, our goal is not to compete with the state-of-the-art multiple objects tracking (MOT) algorithms, but instead to develop a robust application-specific mono-camera based detection, localization, and tracking solution that is efficient and reliable enough to prototype low latency connected safety applications requiring lane level localization accuracy and can be used to evaluate their performance on 5G MEC-based system setup. We are in no way claiming any superiority in terms of accuracy and generalizability over more advanced state-of-the-art tracking algorithms as mentioned in the related work Section II.

Efficacy of super resolution: While scale testing remains a part of our future research agenda, it is contingent upon factors such as public safety considerations and the availability of public 5G and MEC capabilities. In this preliminary work, we capped the maximum variable bitrate to 2 Mbps and mainly examined the impact of limited bandwidth on latency and accuracy metrics, such as distance, speed, and heading estimation. Our findings in these settings demonstrate that lower resolution images can indeed reduce latency compared to higher resolution counterparts while maintaining reasonable accuracy levels. We leave conducting more detailed experiments with various possible bitrates and resolutions, grounded on scene density and complexity, as part of future work.

VII. Conclusion

In this paper, we make the following key contributions. First, we design and implement *Raven*, the first computer vision based connected safety application platform using a real 5G cellular communication and MEC setup. Second, we propose and implement an application specific, practical, and efficient mono-camera based detection, localization, and tracking solution capable of providing lane level or better lateral position accuracy at up to a longitudinal distance of 80 meters, while meeting safety application latency requirements. Third, we implement and evaluate three different connected safety applications namely Intersection T- Bone Vehicle Crash Warning, Vehicle-in-blind-spot Warning and Stopped/Disabled Hazard Vehicle Warning applications on top of *Raven*. Our experiments show promising results: the median position, velocity, and heading accuracy is 2.5 m, 4.9 kph (or 3 mph), and 2.1°, respectively, and the driver is warned in-time during all the test runs of the implemented safety applications. The key scientific value of this work is in demonstrating the possibility of enabling connected safety applications through 5G cellular link and MEC based computing. We hope that the learnings from our implementation will provide a pathway to improve transportation safety. In future we plan to address the problem of more harsh network conditions and scale our system to meet the needs of more common deployment scenarios.

REFERENCES

[1] FCC, "Dedicated short range communications (DSRC) service," 2022, Accessed: Dec. 26, 2022. [Online]. Available: https://www.fcc.gov/wireless/bureau-divisions/mobility-division/dedicated-short-range-communications-dsrc-service

[2] Qualcomm, "Introduction to CV2X," 2019, Accessed: Dec. 26, 2022. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/c-v2x_intro.pdf

[3] P. Tong, M. Li, M. Li, J. Huang, and X. Hua, "Large-scale vehicle trajectory reconstruction with camera sensing network," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 188–200.

[4] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J.-N. Hwang, "Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 108–115.

[5] M. Alleven, "DSRC mandate moving off the table for automakers," 2017, Accessed: Dec. 26, 2022. [Online]. Available: https://www.fiercewireless.com/wireless/report-dsrc-mandate-moving-off-table-for-auto-makers

[6] M. Emara, M. C. Filippou, and D. Sabella, "MEC-assisted end-to-end latency evaluations for C-V2X communications," in *Proc. IEEE 2018 Eur. Conf. Netw. Commun.*, 2018, pp. 1–9.

[7] B. Coll-Perales et al., "End-to-end V2X latency modeling and analysis in 5G networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 5094–5109, Apr. 2023.

[8] A. Kanavos, D. Fragkos, and A. Kaloxylos, "V2X communication over cellular networks: Capabilities and challenges," *Telecom*, vol. 2, pp. 1–26, 2021.

[9] S. Kekki et al., "MEC in 5G networks," *ETSI White Paper*, vol. 28, pp. 1–28, 2018.

[10] F. Giust et al., "MEC deployments in 4G and evolution towards 5G," *ETSI White Paper*, vol. 24, pp. 1–24, no. 2018, 2018.

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

[12] F. Giust et al., "Multi-access edge computing: The driver behind the wheel of 5G-connected cars," *IEEE Commun. Standards Mag.*, vol. 2, no. 3, pp. 66–73, Sep. 2018.

[13] A. G. Guido et al., "Latency assessment of an its safety application prototype for protecting crossing pedestrians," in *Proc. 2020 IEEE 91st Veh. Technol. Conf.*, 2020, pp. 1–5.

[14] K. Antonakoglou et al., "On the needs and requirements arising from connected and automated driving," *MDPI J. Sensor Actuator Netw.*, vol. 9, no. 2, 2020, Art. no. 24.

[15] W. Zhang, M. Feng, M. Krunz, and H. Volos, "Latency prediction for delay-sensitive V2X applications in mobile cloud/edge computing systems," in *Proc. 2020 IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[16] S.-C. Lin, K.-C. Chen, and A. Karimoddini, "SDVEC: Software-defined vehicular edge computing with ultra-low latency," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 66–72, 2021.

[17] P. V. Grammatikos and P. G. Cottis, "A mobile edge computing approach for vehicle to everything communications," *SCIRP Commun. Netw.*, vol. 11, no. 3, pp. 65–81, 2019.

[18] F. Vázquez-Gallego et al., "A mobile edge computing-based collision avoidance system for future vehicular networks," in *Proc. 2019 IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 904–905.

[19] P. L. Nguyen, R.-H. Hwang, P. M. Khiem, K. Nguyen, and Y.-D. Lin, "Modeling and minimizing latency in three-tier V2X networks," in *Proc. 2020 IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[20] L. Bréhon-Grataloup, R. Kacimi, and A. Beylot, "Mobile edge computing for v2x architectures and applications: A survey," *Comput. Netw.*, vol. 206, 2022, Art. no. 108797.

[21] J. Guo, B. Song, S. Chen, F. R. Yu, X. Du, and M. Guizani, "Context-aware object detection for vehicular networks based on edge-cloud cooperation," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5783–5791, Jul. 2020.

[22] R. K. Srinivasa, N. K. S. Naidu, S. Maheshwari, C. Bharathi, and A. R. H. Kumar, "Minimizing latency for 5G multimedia and V2X applications using mobile edge computing," in *Proc. 2019 2nd Int. Conf. Intell. Commun. Comput. Techn.*, 2019, pp. 213–217.

[23] US DOT, "Connected Vehicle Basics," 2015. Accessed: Feb. 15, 2023. [Online]. Available: https://www.its.dot.gov/cv_basics/cv_basics_how_used.htm

[24] P. N. Michelini, Y. Lu, and X. Jiang, "Edge-SR: Super-resolution for the masses," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2022, pp. 1078–1087.

[25] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOV4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[26] D. Martín-Sacristán et al., "Evaluation of Lte-Advanced connectivity options for the provisioning of V2X services," in *Proc. 2018 IEEE Wireless Commun. Netw. Conf.*, 2018, pp. 1–6.

[27] Honda, "Honda and verizon test how 5G enhances safety for connected and autonomous vehicles," Accessed: Apr. 28, 2024. [Online]. Available: https://www.youtube.com/watch?v=KAHPZ7l6qcM

[28] W.-C. Hung, S.-R. Yang, S.-N. Lee, Y.-C. Lin, and P. Lin, "An NFV-based edge platform for low-latency V2X services supporting vehicle mobility-driven auto scaling," in *Proc. IEEE 2021 Int. Wireless Commun. Mobile Comput.*, 2021, pp. 1418–1423.

[29] J. Qin, Y. Xun, and J. Liu, "CVMIDS: Cloud-vehicle collaborative intrusion detection system for Internet-of-Vehicles," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 321–332, Jan. 2024.

[30] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-aware video analytics on edge computing platform," in *Proc. ACM/IEEE Symp. Edge Comput.*, 2017, pp. 1–13.

[31] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proc. 14th USENIX Symp. Netw. Syst. Des. Implementation*, 2017, pp. 377–392.

[32] Q. Liu, S. Huang, and T. Han, "Fast and accurate object analysis at the edge for mobile augmented reality," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, pp. 1–2.

[33] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. 2018 IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 1421–1429.

[34] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. 2018 Conf. ACM Special Int. Group Data Commun.*, 2018, pp. 253–266.

[35] X. Fu, T. Ghaffar, J. C. Davis, and D. Lee, "Edgewise: A better stream processing engine for the edge," in *Proc. 2019 USENIX Annu. Techn. Conf.*, 2019, vol. 19, pp. 929–946.

[36] G. Ananthanarayanan, V. Bahl, L. Cox, A. Crown, S. Nogbahi, and Y. Shu, "Video analytics-killer app for edge computing," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Serv.*, 2019, pp. 695–696.

[37] C. Canel et al., "Scaling video analytics on constrained edge nodes," *Proc. Mach. Learn. Syst.*, vol. 1, pp. 406–417, 2019.

[38] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for -efficient real-time video analytics," in *Proc. ACM SIGCOMM*, 2020, pp. 359–376.

[39] J. He, G. Baig, and L. Qiu, "Real-time deep video analytics on mobile devices," in *Proc. 22nd Int. Symp. Theory, Algorithmic Found., Protocol Des. Mobile Netw. Mobile Comput.*, 2021, pp. 81–90.

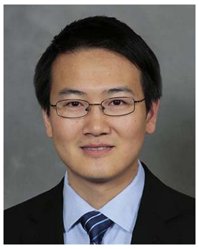
[40] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "Adaframe: Adaptive frame selection for fast video recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1278–1287.

[41] Z. Tang et al., "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8797–8806.

[42] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J.-N. Hwang, "Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 108–115.

[43] M. Brostrom, “SOTA real-time multi-object tracking and segmentation,” 2022. Accessed: Feb. 01, 2023. [Online]. Available: https://github.com/mikel-brostrom/yolov8_tracking
[44] M. Lei, D. Yang, and X. Weng, “Integrated sensor fusion based on 4D MIMO radar and camera: A solution for connected vehicle applications,” *IEEE Veh. Technol. Mag.*, vol. 17, no. 4, pp. 38–46, Dec. 2022.
[45] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *Proc. 2017 IEEE Int. Conf. Image Process.*, 2017, pp. 3645–3649.
[46] Y. Zhang et al., “Bytetrack: Multi-object tracking by associating every detection box,” in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 1–21.
[47] “GStreamer - open source multimedia framework,” 1999. Accessed: Feb. 15, 2023, https://gstreamer.freedesktop.org/
[48] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, “Real-time streaming protocol version 2.0,” no. rfc7826, 2016.
[49] D. Singer, H. Desineni, and R. Even, “A general mechanism for RTP header extensions,” no. rfc8285, 2017.
[50] E. A. Wan and R. V. D. Merwe, “The unscented Kalman filter for nonlinear estimation,” in *Proc. IEEE 2000 Adaptive Syst. Signal Process., Commun., Control Symp.*, 2000, pp. 153–158.
[51] G. F. Basso, T. G. S. D. Amorim, A. V. Brito, and T. P. Nascimento, “Kalman filter with dynamical setting of optimal process noise covariance,” *IEEE Access*, vol. 5, pp. 8385–8393, 2017.
[52] 3GPP, “System architecture for the 5G system. Release-15(v. 2.0.1),” Tech. Rep., Dec. 2017. [Online]. Available: https://www.3gpp.org/specifications-technologies/releases/release-15

Kamran Ali received the B.S. degree in electrical engineering and computer sciences from the School of Science and Engineering, Lahore University of Management Sciences, Lahore, Pakistan, and the Ph.D. degree in computer science and engineering from Michigan State University, East Lansing, MI, USA. He was with Hewlett Packard Labs, NEC Laboratories, Nokia Bell Labs and Microsoft’s Applied Sciences Group during his Ph.D. He is currently a Senior R&D Scientist with General Motors. His expertise and research interests include wireless communications, sensing & control systems, localization & mapping systems, sensor fusion, machine learning, signal processing, mobile/edge computing, and Internet of Things.


Bo Yu received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2007. He is currently a Staff Researcher with General Motors Research and Development, Warren, MI, USA. His research interests include vehicle-to-vehicle (V2V) communications, information dissemination, V2V protocol design, vehicular infotainment, localization and mapping for autonomous driving.

Vivek Vijaya Kumar received the bachelor’s degree in electronics and communications from PESIT, Bengaluru, India, and the master’s degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA. He is currently a Senior Researcher with the Electrical & Controls Systems Research Lab, GM Research & Development. He holds several U.S. patents and has published a number of technical papers. He has more than fourteen years of research experience in use of Communication (DSCR, C-V2X, Cellular) for the development of vehicle safety application. He is a GM Technical Lead for V2X activities at Crash Avoidance Metrics Partnership, and also is a voting member of multiple V2X Technical Committees at SAE.

Krishnan Hariharan (Fellow, IEEE) is currently a Technical Fellow with GM and leads GM R&D programs on connected and autonomous vehicle systems. His work experience includes more than 30 years of overall development of connected vehicle safety and automated vehicle systems for automotive applications. He has made critical, lasting contributions to the research and development of connected vehicle safety systems using vehicle-to-everything (V2X) communications. He has successfully led large cross-OEM and supplier teams under the Crash Avoidance Metrics Partners Consortiums (CAMP) for two decades. He is a Fellow of SAE International. He was the recipient of the prestigious U.S. Government Award for Safety Engineering in 2015.

Fan Bai (Fellow, IEEE) received the B.S. degree in automation engineering from Tsinghua University, Beijing, China, in 1999, and the M.S.E.E. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2005. He is currently a Technical Fellow with the Connected Vehicle Experience Research Lab, Global Research & Development, General Motors. His current research interests include discovery of fundamental principles and the analysis and design of protocols/systems for next-generation vehicular networks, for safety, infotainment and autonomous driving applications. He was the recipient of several GM corporate awards for his contributions to connected vehicle domain. He was also the recipient of four best paper awards or best paper runner-up awards from reputable IEEE/ACM conferences. He is anACM Distinguished Scientist.

Mohammad Rehan received the Ph.D. degree in engineering physics from Embry Riddle Aeronautical University, Daytona Beach, FL, USA. He was also a Postdoctoral Fellow with the University of Central Florida, Orlando, FL, for about two years. He is currently a Senior Software Engineer with Cognizant Technology Solutions. His research interests include dynamics and control of nonholonomic systems, SLAM (simultaneous localization and mapping) and reinforcement learning.

Abhishek Vijay Kumar received the Bachelor of Enginnering degree in electrical & electronics engineering from Manipal University, Manipal, India, in 2009, and the Master of Enginnering degree from the University of Windsor, Windsor, ON, Canada, in 2011. He has overall seven years of experience V2X technologies. He was involved in development of OmniAir certified Conformance Test Tools for V2X-DSRC by Danlaw. He is actively involved in development and testing of DSRC and CV2X vehicle systems.