Proceedings of the IEOM International Conference on Smart Mobility and Vehicle Electrification
Detroit, Michigan, USA, October 10-12, 2023

Vehicle-to-Everything Communication Using a Roadside Unit for Over-the-Horizon Object Awareness

Michael Khalfin
Department of Computational Applied Mathematics and Operations Research
Rice University
Houston, TX 77005, USA
mlk15@rice.edu

Jack Volgren
Department of Engineering
Pennsylvania State University
State College, PA 16801, USA
jqv5334@psu.edu

Luke LeGoullon
Department of Engineering
Louisiana State University
Baton Rouge, LA 70803, USA
llegou1@lsu.edu

Brendan Franz
Department of Computer Science
Harvard University
Cambridge, MA 02138, USA
brendanfranz@college.harvard.edu

Shilpi Shah
Department of Computer Science
Rutgers University
New Brunswick, NJ 08901, USA
sps247@scarletmail.rutgers.edu

Travis Forgach
Department of Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109, USA
tforgach@umich.edu

Matthew Jones
Department of Mathematics
Willamette University
Salem, OR 97301, USA
mpjones@willamette.edu

Proceedings of the IEOM International Conference on Smart Mobility and Vehicle Electrification
Detroit, Michigan, USA, October 10-12, 2023

Milan Jostes, Ryan Kaddis, and Chan-Jin Chung
Department of Math and Computer Science
Lawrence Technological University
Southfield, MI 48075, USA
mjostes@ltu.edu, rkaddis@ltu.edu, cchung@ltu.edu

Joshua Siegel
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
mpjones@willamette.edu

Abstract

Self-driving and automated vehicles rely on a comprehensive understanding of their surroundings and one another to operate effectively. While the use of sensors may allow the vehicles to directly perceive their environments, there are instances where information remains hidden from a vehicle. To address this, vehicles can transmit information between each other, enabling over-the-horizon awareness. We create a Robot Operating System simulation of vehicle-to-everything communication. Then, using two real-life electric vehicles equipped with global positioning systems and cameras, we aggregate time, position, and navigation information into a central database on a roadside unit. Our model uses an image classification deep learning model to detect obstacles on the road. Next, we create a web-based graphical user interface that automatically updates to display the vehicles and obstacles from the database. Finally, we use an occupancy grid to predict vehicle trajectories and prevent potential collisions. Our deep learning model has a precision-recall score of 0.995 and our system works across many devices. In the future, we aim to recognize a broader range of objects, including pedestrians, and use multiple roadside units to widen the scope of the model.

Keywords
V2X Communication, Occupancy Grid, Over-the-Horizon Data, Trajectory Prediction and Speed Limit Control.

1. Introduction

To combat the eighth leading cause of death globally and the leading cause of death in the United States of America for ages 1-54 (CDC 2023), various parties, including researchers, concerned guardians, government officials, and scientists, have been actively involved in enhancing road quality and safety measures. This includes implementing more speed checks to ensure compliance with speed limits, fostering interactions with law enforcement (IEEE 2023), and advancing autonomous self-driving technology.

One significant technology that has emerged to bolster road safety is Vehicle-To-Everything (V2X) communication (Siegel et al. 2017). V2X enables wireless connection between and among vehicles and other road elements such as pedestrians, cones, buildings, and signs. V2X communication creates an accurate digital representation of the road by integrating data from several vehicles into one cohesive environment (Zhou 2020).

We develop a scalable V2X communication system with the capability to predict and display various vehicular risks, including collisions, various objects, and traffic conditions. We implement this model in two modified Polaris Gem e2 electric vehicles, referred to as Autonomous Campus Transport (ACTor) vehicles: ACTor 1 and ACTor 2. These vehicles are sponsored by MOBIS, US Army GVSC, NDIA, DENSO, SoarTech, Realtime Technologies, Veoneer, Dataspeed, GLS&T, and LTU. Each vehicle is equipped with essential components for automated driving, including a drive-by-wire system, Mako G-319 Camera, 2-dimensional and 3-dimensional Light Detection and Ranging (LiDAR) sensors, and two Swift Piksi Multi Global Navigation Satellite System Modules. The Swift Piksi module uses multiple satellite constellations to acquire positioning data that is accurate to 1 centimeter. For our research, we utilize Latitude, Longitude, and Heading (LLH) values provided by the Swift Piksi modules. Both ACTor 1 and ACTor 2 are equipped with computers to process this data.

As both vehicles are equipped to collect data from their environment, our goal is to create a model that allows them to communicate these data between one another via an intermediary capable of storing and observing data. Utilizing a roadside unit (RSU), both vehicles are able to establish a secure connection to a central server where information can be received, stored, and disbursed. Our RSU consists of a Raspberry Pi 4 Model B minicomputer, 12V battery, and 12V-5V adapter, as seen in Figure 1. In the proceeding sections of this paper, we discuss the connectivity between our vehicles and the RSU, highlighting its scalability. Additionally, we emphasize the aggregation, analysis, display, and transmission of information regarding object awareness and trajectory planning. Beyond simulation, we conduct our experimentation and evaluation process at Lawrence Technological University (LTU) Parking Lot H located in Southfield, Michigan, which provides a suitable two-lane course.

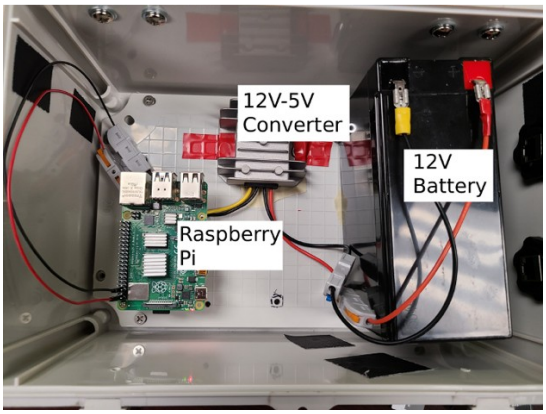

Figure 1. Roadside Unit Setup

To meet our goals of effective communication, we propose a V2X communication system containing the RSU, object detection model, and a database to transmit information from ACTor 1 to the Roadside Unit (RSU) to ACTor 2 and also vice-versa using Wireless Access in Vehicular Environments (WAVE) technology. This model receives visual information from the camera in the vehicle to detect orange traffic cones. We also predict a trajectory for each vehicle within an ensured remote speed limit. We then use this information to populate an occupancy grid, displaying both vehicles, their predicted trajectories, and detected objects. Finally, we program and host a web graphical user interface (GUI) to display live-updating markers visually for the vehicles and detected objects. The culmination of these various elements ensures the reliability of a comprehensive V2X system.

2. Literature Review

V2X technology has allowed for more rapid and wide-spread development of Intelligent transportation systems (ITS) \cite{Chen et al. 2017} to allow for an interconnected society by wireless network. ITS can therefore predict and adapt to dangerous situations, dynamic events, and traffic management (Shaheen and Finson 2013}. V2X is projected to be highly important in the future of roadways (Zhou et al. 2020). It is comprised of vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-network communication (V2N) (MacHardy et al. 2018}.

V2V communication involves the transmission of information between vehicles. This communication enables vehicles to exchange data related to collision detection and congestion reduction. Dedicated Short-Range Communications (DSRC) technology is commonly used for secure and reliable information transmission in V2V communication. DSRC offers features such as high accuracy, low latency, and specific bandwidth allocation for efficient V2V communication (Zorkany et al. 2020).

DSRC is moreover utilized in V2I communication, which primarily focuses on traffic management. A WAVE system is commonly employed for V2I communication. The WAVE system consists of three main components: an RSU, an onboard unit to ensure stable connections between vehicles and infrastructure, and service channels for bidirectional communication (Milanes et al. 2012}. Through V2I communication, vehicles can share and gather data from their surroundings, including road layout information and proximity between vehicles. This enables them to avoid collisions and anticipate potential traffic risks (Milanes et al. 2012). V2V and V2I communication are often closely integrated in the development of ITS.

Lastly, V2N communication is the connection between a vehicle and a server. The coordination of V2N and V2I communication systems usually allows for safe and more accurate development of a V2V communication system (Dein et al. 2020). (10 font)

3. Methods

We begin by explaining our virtual simulation of V2X technology. Then, we shift our focus to a real-time system with the RSU and ACTor vehicles. We first overview the software architecture. Afterwards, we expand on the individual software components.

3.1 Gazelle Simulation

Access to the hardware necessary for development and testing, like the ACTor vehicles, is not always available. In order to test new implementations of V2X and other software systems quickly and efficiently, aspects of the project are modeled in a simulated environment. Using Python and Robot Operating System (ROS) (ROS 2023), two vehicles are simulated on the selected experimental test course in the simulator Gazelle. Gazelle is a 2-dimensional lightweight vehicle simulator in development at LTU. The simulator is designed to reflect the architecture and usage of the ACTor vehicles, including sensors like a camera, LiDAR, and GPS receiver. It supports the simulation of multiple vehicles, making it ideal for this project. Two vehicles are simulated to follow a lane on the Lot H test course and report their relevant information, like position and heading (odometry), to the virtual RSU, which handles collision avoidance and correction.

Each simulated vehicle is comprised of three ROS nodes and their interactions, as shown in Figure 2. One node is responsible for vehicle steering, which is done by detecting and responding to the lane lines. The node then publishes its calculated yaw value to be received by the controller node. The controller node handles vehicle movement and communication with the RSU. When the RSU predicts a crash it informs the controller node, and the vehicle responds accordingly. The final node is responsible for detecting the red marker on the outside of the lane, which acts as a lap indicator. The RSU node collects each vehicle's GPS location and odometry information and determines whether a crash is about to happen. If two vehicles are on course to collide, the RSU stops the vehicle that would collide into the other for a moment, before allowing it to move again. This node, in conjunction with the controller nodes of active vehicles, effectively prevents and corrects collisions before they happen.

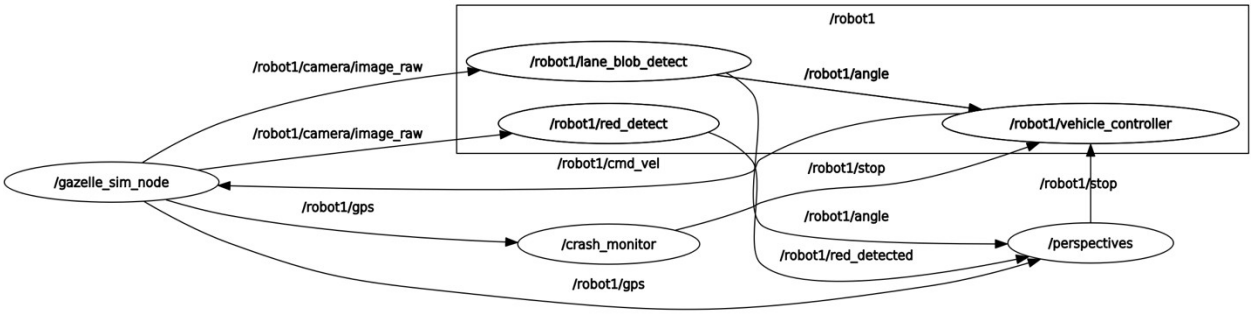

Figure 2. ROS Node architecture of a single vehicle in the V2X Gazelle Simulation

In Figure 3, we show a snapshot of the simulation in action, with the two vehicles depicted in green and blue. We also display the sample calculations for collision detection in the bottom left corner.

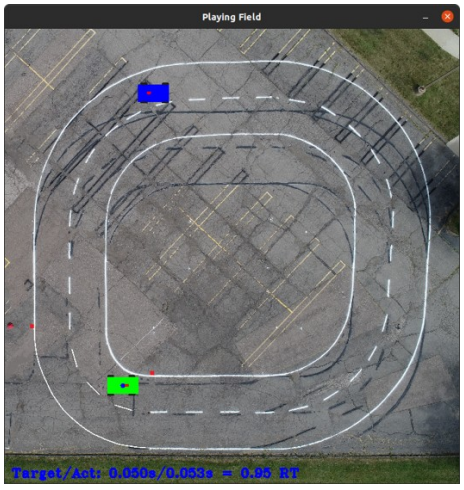

Figure 3. Aerial View of LTU's Lot H Test Course in Gazelle.

3.2 Real-Time Vehicle System

We enable remote access on the Raspberry Pi. We use Secure Shell Protocol (SSH) for remote login and command-line execution.

Our devices are connected to the RSU Wireless Fidelity (Wi-Fi) network. We give the RSU an internet connection as well as an external Wi-Fi adapter to set up a hot spot. Devices then connect to the RSU through the hotspot.

We use ROS for our robot software development needs and for streamlined communication between devices. ROS architecture uses a publisher-subscriber system, so that nodes can publish, or send, messages to a topic (ROS 2023). Alternatively, they can subscribe to, or receive, messages from a topic for further use in their scripts.

Our ROS Master Uniform Resource Identifier comes from the RSU. All devices in the network use their own Internet Protocol (IP) addresses as ROS host names. We also use C++ and Python to create custom ROS messages, source files, and scripts.

3.2.1 Software Architecture

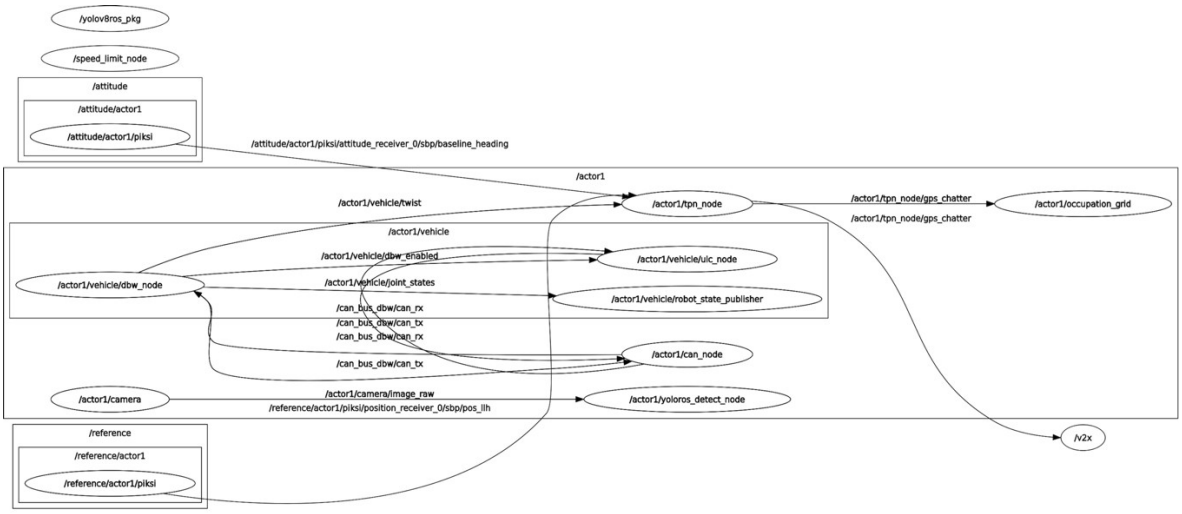

Figure 4. ROS Node architecture of the real-time vehicle system

Our ROS architecture with our nodes and topics is shown in Figure 4. We elaborate on the most important aspects of the architecture in the following paragraphs.

We create a Time, Position, and Navigation (TPN) node which subscribes to the Latitude, Longitude, and Heading (LLH) topic from the Piksi Global Positioning System (GPS) and the Vehicle Twist topic from the drive-by-wire system. Through a system of callback functions, the C++ program packages the current LLH, linear velocity, and yaw rate of each vehicle as a class, which is then converted to a custom ROS message type, called Position, upon publishing. We show sample C++ code to accomplish this task in Figure 5. We publish this message to the ROS topic named GPS Chatter to the RSU. This can be used to accurately live-update the position of the vehicle and calculate an estimated trajectory using the heading and velocity parameters of the message.

```
void HelloTPN::positionCallback(const libsbp_ros_msgs::MsgPosLlh::ConstPtr& msg)
{
    libsbp_ros_msgs::MsgPosLlh llh = *msg;
    this->pos.lat = llh.lat;
    this->pos.lon = llh.lon;
    ROS_ERROR_STREAM("positionCallback -- lat: " << lat << " lon: " << lon);
}

void HelloTPN::headingCallback(const libsbp_ros_msgs::MsgBaselineHeading::ConstPtr& msg)
{
    libsbp_ros_msgs::MsgBaselineHeading baseline_heading = *msg;
    this->pos.heading = baseline_heading.heading / 1000;
    ROS_ERROR_STREAM("headingCallback -- heading: " << heading);
}

void HelloTPN::velocityCallback(const sensor_msgs::Twist::ConstPtr& msg) {

    this->pos.velocity = msg.linear.x;
    ROS_ERROR_STREAM("velocityCallback -- velocity: " << velocity);
}
```

Figure 5. Packaging LLH and velocity data inside TPN node

Additionally, we train a You Only Look Once Version 8 (YOLOv8) (Jocher et al. 2023, Kaddis et al. 2023) cone detection model based on Blue-Green-Red (BGR) images captured using the Camera Raw BGR Image node. Then, we develop a YOLOv8 ROS Package with a cone detection node that subscribes to the Raw Camera BGR Image topic. It subsequently publishes a list of boundary boxes around the cones and confidence values using a custom message type called BBoxes. Afterwards, we publish this message to the Predictions topic making the message visible to the RSU and all of the vehicles connected to the network.

On the Raspberry Pi, we develop a V2X database using the MariaDB client-server. It has two tables: Client Vehicles and Identified Objects. We create a V2X node which subscribes to the GPS Chatter and Predictions topics, and then updates the database with the most recent vehicle and cone information. Additionally, we use Node.js, HTML, Socket.IO, and Google Maps Application Programming Interface (API) to generate a web-based GUI, which automatically updates based on information from the database.

We also create a remote speed limit node on the RSU that publishes a Float32 value to the Speed topic. Finally, in our vehicles, we create a Trajectory Prediction package, which has nodes that subscribe to the GPS Chatter, Predictions, and Speed topics. It ultimately uses these topics to predict vehicle trajectories and generate occupancy grids of vehicles and objects on the test course.

3.2.2 Time, Position, and Navigation

Although we use two vehicles, our TPN node is designed to be scaled to many vehicles on the road. In our TPN node launch files, we prefix our GPS Chatter topic names with "Actor1" and "Actor2" to create topics for each vehicle to publish their unique Position data to the RSU. Each vehicle needs a designated numerical ID in order to maintain accurate vehicle predictions and maintains the database's structure.

3.2.3 Cone Detection Model

At 3:00 PM on a moderately cloudy day, we capture 312 images with cones and 100 images without cones. We use RoboFlow to manually identify the cones, or lack thereof, in each image. We also use to RoboFlow to augment our images by randomly applying rotation of up to 45 degrees in either direction and adding noise. Next, we randomly create a data set with our original and augmented images with a 70% training, 15% testing, and 15% validation split. We train our model with the YOLOv8 small model over 100 epochs, as seen in Figure 6.

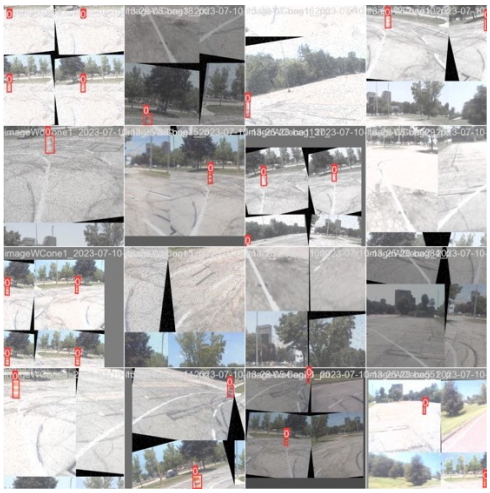

Figure 6. Training using the YOLOv8 small model

With consistently cloudy weather conditions, our model has 85-95% confidence that it detects cones. We obtain this value over 50 trials using YOLOv8's built-in threshold confidence, designed to eliminate false negatives (Jocher et al. 2023). In rainy weather conditions, our model has 40-60% confidence. Lastly, our model detects cones with ~80% confidence after sunset. Sample output from our model in normal, rainy, and dark conditions is displayed in Figure 7.

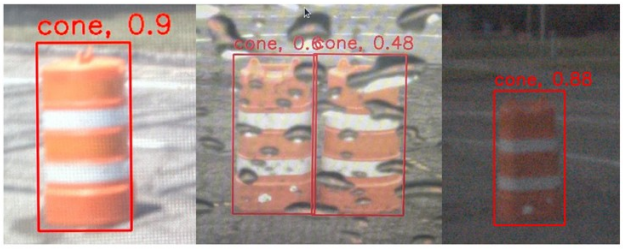

Figure 7. Sample cone detection in broad daylight, rainy weather, and after sunset

3.2.4 Data Aggregation and Visualization

We configure a cursor through the PyMySQL module in Python to update the database in real-time through a ROS program.

The Client Vehicle Table uses a unique index for each vehicle. These are the same indices from the TPN designation. The table stores the most recent latitude, longitude, heading, speed, and angular velocity (yaw rate) values per vehicle.

On the other hand, the Identified Objects Table generates a unique index for each object detected. It stores the 30 most recent objects detected using a queue to add and remove objects as necessary. The table stores latitude, longitude, type of object detected, and the time that the object was detected.

We configure the database to support remote hosting. Then, any vehicle in the RSU network with access to the name, username, and password may connect to the database.

For the GUI web server, we pull information from the database every 500 milliseconds. We encode ACTor 1 and ACTor 2 with blue and green markers respectively, and objects with orange markers. Markers are overlaid on a map with satellite footage from Southfield, MI. Sample output during testing is in Figure 8. The website is temporarily hosted with the Raspberry Pi's local host on Port 3000. We are able to view the website in real-time on web browsers during the demos.

Figure 8. ACTor vehicles and a cone object plotted with Google Maps API

3.2.5 Trajectory Prediction and Visualization

The role of the occupancy grid is to manage intersections. We plot vehicles connected to the RSU and predict their paths using a simplified version of the Bicycle Model (MathWorks 2023) with inputs for velocity and steering angle. By calculating the trajectories of various vehicles, the occupancy grid can predict, display, and prevent potential collisions. An image of the occupancy grid is displayed in Figure 9. The green squares represent sections of the grid occupied by vehicles, while the yellow square represent their trajectories. Finally, the red squares signify potential collisions.

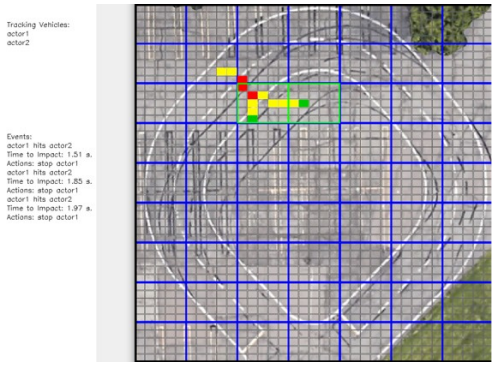

Figure 9. Occupancy grid displaying two vehicles in close proximity

We integrate the remote speed limit to ensure vehicles are driving within designated speed limits.

5. Results and Discussion

5.1 Demonstration

As mentioned earlier, Figure 9 depicts an example of the occupancy grid in use. Two vehicles are connected to the network and populated on the grid. Their paths are predicted using the velocity and steering angle that they report using the TPN message. A collision is predicted for every square in which the predicted trajectories overlap. Once a collision is predicted, the system displays simple resolving commands to prevent it. To reserve computational power for other tasks, vehicles are resolved by time rather than space, meaning the slower moving vehicle is halted, yielding to the other vehicle. The occupancy grid differentiates between the vehicles and obstacles. When a collision is detected between a vehicle and an obstacle the vehicle is brought to a stop. This allows each vehicle to use its onboard capabilities to avoid the obstacle.

We demonstrate our web GUI and occupancy grid in real-time. Here is a video from demonstration day.

5.2 Quantitative Results

We only trained our YOLOv8 model on cones. It detects cones well. The precision-recall score for our model is 0.995. Our recall-confidence score is approximately 1.0 for a confidence value less than 0.8. Then, it starts to decrease rapidly to 0.0 at 1.0 confidence. We show these metrics and more in Figure 10. These are the graphs generated by YOLOv8 evaluating our model's performance.

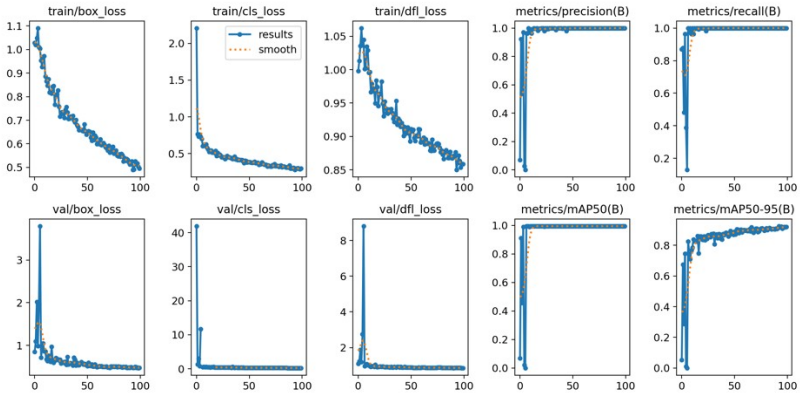

Figure 10. Various graphs displaying the success of the deep learning model

5.3 Discussion

Using a local hot spot and Wi-Fi, we can facilitate communication between devices through internet connection to enable our network. However, to expand our implementation, we will require cloud-based Internet of Vehicles technology for big data storage, fast processing, and wider scalability (Zhou et al. 2020).

Like Condoluci et al. (2023), we use edge computing, as we do all the calculations related to the occupancy grid on the vehicles themselves. On the other hand, the goal of the researchers Condoluci et al. (2023) was to improve cellular 5G V2X communication. However, we opt to demonstrate a proof-of-concept V2X software architecture on our test course without describing how to connect and disconnect from Wi-Fi on the road.

5.4 Limitations and Future Work

Vehicles encounter obstructions including potholes, road debris, animals, pedestrians, and other vehicles. Although we train a YOLOv8 small model to detect cones, we would ultimately like to implement a Segment Anything Model (Kirillov et al. 2023}.

Moreover, we recognize that static obstacles, such as potholes, may stay on the road for several years. On the other hand, dynamic obstacles, such as animals or pedestrians, may only be on the road for a few seconds. Instead of a FIFO data structure, we should eliminate rows from our Object Detection Table based on the amount of time that passes, varying the amount of time based on the type of object detected. We can also have vehicles with the same LLH coordinates in the future verify if the object in the database is still there.

Another limitation of our project is that we use one RSU, whereas a stable V2X system should have multiple RSU's. This way, if one encounters a security breach or runs out of power, there are other RSU's which can take over. Our project does not consider threats to security as a whole, nor do we evaluate privacy risks. These are all important areas to focus on in the future. (10 font)

6. Conclusion

We demonstrate and evaluate a V2X communication system for object awareness and trajectory planning between two vehicles and a RSU, consisting of a Raspberry Pi and a battery. To accomplish this objective, we build a virtual model which has multiple vehicles engaging in V2X communication. For our real-time vehicle system, we create a temporary Wi-Fi hot spot to broadcast TPN and object detection messages to the Raspberry Pi. The Raspberry Pi aggregates the information in a database to show the objects of interest in a web browser using Google Maps API. We also implement an occupancy grid to anticipate vehicle trajectories and collisions. We successfully show that a V2X system can enhance road safety by using sensors from distributed vehicles on the road to pool over-the-horizon information.

References

Condoluci, Massimo, Gallo, Laurent, Mussot, Laurent, Kousaridas, Apostolos, Spapis, Panagiotis, Mahlouji, Maliheh, and Mahmoodi, Toktam. 2019. 5G V2X System-Level Architecture of 5GCAR Project. Future Internet. https://doi.org/10.3390/fi11100217.

Deinlein, Thomas, German, Reinhard, and Djanatliev, Anatoli. 2020. 5G-Sim-V2I/N: Towards a simulation framework for the evaluation of 5G V2I/V2N use cases. European Conference on Networks and Communications (EuCNC), 353–357. https://doi.org/10.1109/EuCNC48522.2020.9200949.

El Zorkany, Mohamed, Yasser, Ahmed, and Galal, Ahmed I. 2020. Vehicle to vehicle "V2V" communication: scope, importance, challenges, research directions and future. The Open Transportation Journal, 14, 86–98. https://doi.org/110.2174/1874447802014010086.

Jocher, Glenn, Chaurasia, Ayush, and Qiu, Jing. 2023. YOLO by Ultralytics. Retrieved from https://github.com/ultralytics/ultralytics.

Kim, Jeong ah, Sung, Ju-Yeong, and Park, Se ho. 2020. Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. In 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 1–4. https://doi.org/10.1109/ICCE-Asia49877.2020.9277040.

Kirillov, Alexander, Mintun, Eric, Ravi, Nikhila, Mao, Hanzi, Rolland, Chloe, Gustafson, Laura, Xiao, Tete, Whitehead, Spencer, Berg, Alexander C., and Lo, Wan-Yen. 2023. Segment Anything. arXiv:2304.02643. https://arxiv.org/abs/2304.02643.

Lumelsky, Vladimir J. 2004. Algorithmic and complexity issues of robot motion in an uncertain environment. Journal of Complexity, 3(2). https://doi.org/10.1016/0885-064X(87)90025-2.

MacHardy, Zachary, Khan, Ashiq, Obana, Kazuaki, and Iwashina, Shigeru. 2018. V2X access technologies: regulation, research, and remaining challenges. IEEE Communications Standards & Tutorials, 20(3), 1858–1877. https://doi.org/10.1109/COMST.2018.2808444.

Milanes, Vicente, Villagra, Jorge, Godoy, Jorge, Simo, Javier, Perez, Joshué, and Onieva, Enrique. 2012. An intelligent V2I-based traffic management system. IEEE Transactions on Intelligent Transportation Systems, 13(1), 49–58. https://doi.org/10.1109/TITS.2011.2178839.

Shaheen, Susan, and Finson, Rachel. 2013. Intelligent Transportation Systems. UC Berkeley: Transportation Sustainability Research Center. https://escholarship.org/uc/item/3hh2t4f9.

Siegel, Joshua E., Erb, Dylan C., and Sarma, Sanjay E. 2017. A survey of the connected vehicle landscape—Architectures, enabling technologies, applications, and development areas. IEEE Transactions on Intelligent Transportation Systems, 19(8), 2391–2406.

IEEE Public Safety Technology. 2023. Advances in Road Safety Technology. Retrieved July 16, 2023, from https://publicsafety.ieee.org/topics/advances-in-road-safety-technology.

Zhou, Haibo, Xu, Wenchao, Chen, Jiacheng, and Wang, Wei. 2020. Evolutionary V2X technologies toward the internet of vehicles: challenges and opportunities. Proc. IEEE, 108(2), 308–323. https://doi.org/10.1109/JPROC.2019.2961937.

Additional Information

The source code for this paper can be found in the following Github repositories:

- https://github.com/michael-khalfin/tpn_pkg
- https://github.com/michael-khalfin/yolov8ros_pkg
- https://github.com/michael-khalfin/v2x_pkg
- https://github.com/michael-khalfin/v2x_pkg-pi
- https://github.com/michael-khalfin/v2x_website

Acknowledgements
Our work was supported by the National Science Foundation under Grant Nos. 2150292 and 2150096. We thank our mentors, Joe DeRose and Nick Paul, as well as our teaching assistant, Justin Dombecki.

Biographies

Michael Khalfin is a second-year student at Rice University pursuing a dual degree in Mathematics and Operations Research. His research focused primarily on setting up, maintaining, and managing queries to the MariaDB database on the Raspberry Pi. Michael oversaw the remote connectivity of the devices via the Robot-Operating System and live demonstration. Michael's research interests include optimization methods, mathematical models, and cryptography. In his free time, he enjoys competitively playing chess, swimming, cycling on local trails, and hiking.

Jack Volgren is a Computer Engineering student at The Pennsylvania State University. His studies cover hardware and software engineering. His research has centered around self-driving vehicles and artificial intelligence.

Luke LeGoullon is a third-year student at Louisiana State University pursuing a dual degree in Computer Science and Mathematics. His research focused primarily on efficiently capturing vehicle data and implementing seamless transmission methods. Luke played a pivotal role in synchronizing various components of the project and conducted field testing to ensure its success.

Brendan Franz is a current junior at Harvard University working toward a double concentration in Computer Science and Economics. He is originally from Lake Orion, Michigan and grew up around the auto industry. Currently, Brendan serves as the Director of Technology for a Harvard-based Nonprofit. In his free time, Brendan enjoys golfing, spending time with friends, watching sports, and listening to music.

Shilpi Shah is a sophomore at Rutgers University studying Computer Science and Cognitive Science, with a minor in Mathematics. She helped develop and test the Web GUI and conducted the literary analysis and external research for this paper. This is an academic subject of great interest to her as she aspires to pursue a graduate degree in deep learning quantum computing with planetary applications.

Travis Forgach is a junior at the University of Michigan. Travis is pursuing a Bachelor's Degree in Computer Science with a minor in Statistics. His research interests are artificial intelligence, mobility, machine learning and algorithm development.

Matthew Jones is a student at Willamette University majoring in Mathematics and Computer Science. He is scheduled to graduate in the spring of 2024. He is currently working as a mathematics tutor at his university and has been for 2 years. The areas of research he is interested in are machine learning, optimization, combinatorics, and cybersecurity.

Milan Jostes is a student at Lawrence Technological University studying Computer Science with a focus on Scientific Software. He was awarded the Intel Excellence in Computer Science award twice and became a Scholar of Distinction in 2020. Milan created his own security system and attended the International Science and Engineering Fair (ISEF) as both an observer and presenter for his research. He hopes to achieve his master's degree after four years of college and is considering the pursuit of a doctorate after graduation.

Ryan Kaddis is an undergraduate student at Lawrence Technological University in the Computer Science program. He was an assistant and mentor for the NSF REU at LTU in 2023. He is an active member on LTU's team for the Intelligent Ground Vehicle Competition (IGVC), a design and programming competition for automated vehicles. Ryan's research experience is primarily concerned with autonomous robotics and computer vision.

Chan-Jin Chung, PhD is a Computer Science professor with expertise in autonomous mobile robotics, evolutionary computation, evolutionary-neuro-fuzzy algorithms, deep learning, and STEM+CS education. He was a senior research scientist at Electronics & Telecommunications Research Institute in Korea where he was involved in developing a digital switching system that later became the base system for the first commercialized CDMA system in the world. His doctoral research at Wayne State University was the development of a self-adaptive AI framework motivated by cultural evolution processes, which was then applied to solve various optimization problems. He launched Robofest,

an autonomous robotics competition and mentored college robotics teams for IGVC, RoboCup, and World Robot Olympiad. In 2011, IEEE honored Dr. Chung with its citation of honor award for his leadership in STEM education. His current projects include self-drive algorithm development using by-wire vehicles funded by the National Science Foundation.

Josh Siegel, PhD is an Assistant Professor of Computer Science and Engineering at Michigan State University and the lead instructor of the Massachusetts Institute of Technology's DeepTech Bootcamp. He received Ph.D., S.M. and S.B. degrees in Mechanical Engineering from MIT. Josh and his automotive companies have been recognized with accolades including the Lemelson-MIT Student Prize and the MassIT Government Innovation Prize. He has multiple issued patents, published in top scholarly venues, and been featured in popular media. Dr. Siegel's ongoing research develops architectures for secure and efficient connectivity, applications for pervasive sensing including vehicle diagnostics, and new approaches to automated driving.