2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

Digital Twin for Pedestrian Safety Warning at a Single Urban Traffic Intersection

Yongjie Fu[†], Mehmet K. Turkcan[‡], Vikram Anantha[§], Zoran Kostic[‡], Gil Zussman[‡], and Xuan Di[†*]
[†]Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, NY, 10027.
[‡]Department of Electrical Engineering, Columbia University, New York, NY, 10027.
[*]Data Science Institute, Columbia University, New York, NY, 10027.
[§]Lexington High School, Lexington, MA, 02421.

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588544

***Abstract*—Ensuring the safety of Vulnerable Road Users (VRUs) at intersections is crucial to enhancing urban traffic systems. This paper introduces a novel intelligent warning system specifically designed to increase the safety of VRUs crossing intersections. The proposed system leverages the COSMOS testbed to obtain real time vehicle information and employs Message Queuing Telemetry Transport (MQTT) as a standards-based messaging protocol for device communication and data transmission and utilizes a transformer model and Time To Collision (TTC) method to predict the collision. To validate the effectiveness and reliability of our intelligent alert system, we conducted comprehensive tests using the CARLA simulator, incorporating hardware in the loop simulation approach. The results demonstrate the potential for increased situational awareness and reduced risk factors associated with VRUs at intersections. Our work supports the integration of this intelligent alert system as a viable solution for reducing accidents and enhancing the overall safety of urban intersections in real time.**

***Index Terms*—Digital twin, Pedestrian safety warning, Urban transportation intersection, Transformer, Internet of things**

I. Introduction

Intersections, where forty percent of crashes happen [1], are bottlenecks of urban roads. Recent years have seen a surge in pedestrian fatalities at urban intersections, further aggravated by the emergence of the COVID-19 pandemic.

Metropolises such as NYC have many traffic intersections surrounded by high-rise buildings. Vehicles and pedestrians approaching an intersection have a severely restricted view, caused by buildings, vehicles, and pedestrians. It is challenging to manage urban traffic with multi-modal intersections, which include passenger cars, trucks, buses, pedestrians, and cyclists. Online traffic management requires close monitoring and prediction of each road user's dynamic location/speed/orientation and accurate AI-based decision-making. The emergence of smart city technology [2]–[4] holds great potential to reduce road accidents and improve urban travel experiences.

Applications relevant to traffic safety play a key role in urban traffic management. To achieve this goal, urban digital twin is a tool that leverages cutting-edge sensing, communication, computing, and automation technology. Digital twin is the digital replica of a physical entity in the real world [5].

This paper proposes an integrated system that identifies objects and employs a transformer-based prediction model to anticipate their movements. This system distributes anonymous information to a public server. Specifically, it broadcasts MQTT messages wirelessly, conveying data about timestamps, locations, and types of road users. The user's phone transmits its location to the server using WiFi or 5G. Subsequently, the server calculates potential collision risk information by computing the Time To Collision (TTC) and issues a warning to the user if the computed TTC falls below a predetermined threshold.

II. Related Work

The majority of literature on digital twin for transportation is focused on vehicular Applications on highways, where a digital twin runs on an in-vehicle computer to collect historical driving data and optimize or instruct future maneuvers based on sensing information from the environment. Urban traffic, however, involves many heterogeneous agents that interact rapidly in a short time period. Among these agents, the majority of them are vulnerable road users like pedestrians, cyclists, and macro-mobility users. Their presence and behaviors that are hard to predict pose challenges in modeling and simulation of urban traffic environments with digital twins.

Infrastructure based sensors include street-level cameras, LiDARs, vehicle loop detectors, pedestrian detectors, and location beacons. Bird's-eye cameras mounted on high-rise building capture the totality of traffic in an intersection, providing the overall situational awareness. These sensors are connected to the edge cloud and cloud via wireless and fiber. Street-level cameras or mobile sensors extract individual mobile traces with a higher frequency. We argue that rather than relying on new technology such as DSRC, we would like to leverage existing infrastructure for system development and deployment.

In preliminary work [6]–[8], we focused on the detection and tracking of objects (pedestrians and vehicles) based on deep-learning techniques, modified with the goal of achieving safety-critical accuracy and real time performance (with the

[*]*Corresponding author: Xuan Di, email: sharon.di@columbia.edu.*
This work is sponsored by NSF CPS-2038984, NSF ERC-2133516 and NSF CNS-1827923.

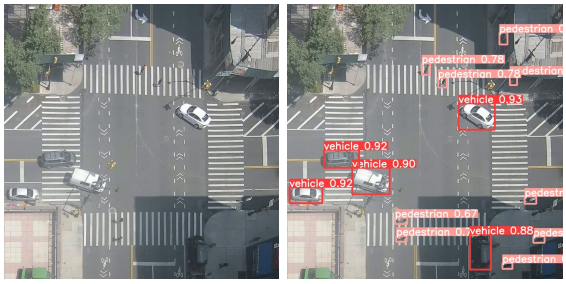

Fig. 1. Sample object detection results from the high-altitude camera deployment enabling the Application. Left - Raw image output retrieved from the camera. Right - Object detection results.

target of 30 frames per second (fps) for densely populated intersections). Preliminary results on the dataset from the COSMOS testbed, using bird's eye cameras only, are reported in [6], with recent improvements in achieving higher fps.

There are several existing works focusing on vehicle [9] or pedestrian safety protection at intersections. Tahmasbi [10] proposes a system that uses Dedicated Short-Range Communications (DSRC) to facilitate communication between vehicles and Vulnerable Road Users (VRUs), employing an "unavoidable crash zone" concept to predict possible collisions. Similarly, Arena [11] introduces a system enabling smartphones to communicate with the onboard units (OBUs) in vehicles via Wi-Fi; this system equips both vehicles and smartphones with collision estimation modules. Gelbal [12] develops a system that utilizes Bluetooth for communication between VRUs and vehicles, also incorporating collision zone predictions.

Regarding the novelties in our system, we utilize the MQTT protocol, which supports both Wi-Fi and 4G/5G connections. The prediction of collisions is calculated on the server, ensuring fast computation and the capability to handle multiple users. Additionally, we employ a transformer-based model to predict the trajectories of both vehicles and pedestrians, which aids in calculating the TTC.

The rest of this paper is organized as follows. Section III introduces the COSMOS testbed we use for the experiments. Section IV introduces the methodology used in this work. Section V explains the system design for the pedestrian warning Application. Section VI concludes our work.

III. Experimentation Testbed

In this section, we will introduce how we collect, aggregate, and process data arriving from sensors at smart intersections using low-latency wireless and GPU-enabled edge cloud node. The data is passed to the AI-powered system for real-time smart intersection object detection and tracking.

A. COSMOS testbed

We will use the COSMOS testbed as a platform for both determining design constraints, corresponding to future wireless networks, and for experimentation and evaluation. The NSF PAWR COSMOS ("Cloud enhanced Open Software defined mobile wireless testbed for city-Scale deployment") project is aimed at the design, development, and deployment of an advanced wireless city-scale testbed in order to support real-world experimentation on next-generation wireless technologies and Applications [13]. The testbed is being deployed in West Harlem, NYC, next to Columbia University. The testbed targets the technology "sweet spot" of ultra-high bandwidth and ultra-low latency, a capability that will enable a broad new class of Applications, including CAVs and smart intersections. Realization of such Applications involves not only faster radio links, but also aspects such as spectrum use, networking, and edge computing.

Edge cloud technology is integrated into COSMOS and is expected to support low-latency Applications. The testbed has been equipped with sensors, including cameras and LiDARs at intersections. Specifically, Fig. 1 illustrates the COSMOS pilot site with cameras overlooking the intersection of Amsterdam Ave. and 120th St., and with high speed (fiber) networking and edge-cloud resources.

IV. Methodology

A. Multi-objective detection

To support the low-latency use cases at high reliability levels needed for smart intersections, we choose high-altitude camera deployments, for which current pre-trained models in commonly used datasets like COCO are not suitable. For object tracking, we use a custom hand-annotated dataset and train models using the single-stage YOLO detector [6]. Tracking is enabled through the ByteTrack, a multi-object tracking (MOT) approach that supports our low-latency use case [14]. Projecting high-altitude camera views to a low-resolution birds-eye view with 832x832 resolution using a hand-crafted perspective transformation, we perform object detection without slicing the input frame into sub-windows.

B. Multi-objective transformer prediction

Obtaining frames from cameras and running object detection and tracking algorithms on the camera frames introduces a delay that might be in the order of seconds. In order to deal with this issue, trajectory forecasting methods can be employed. After trajectory forecasts are obtained, intersections between forecasted curves could be checked to predict the possibility of collision and thus raise safety warning signals.

We employ a framework analogous to the one described in Giuliari's work [15], wherein each object is modeled by an instance of the transformer network. For each object i, we observe a set $F_{\text{obs}} = \left\{x_t^{(i)}\right\}_{t=-(T_{\text{obs}}-1)}^{0}$, which is the current and prior positions of coordinates. We need to predict $F_{\text{pred}} = \left\{x_t^{(i)}\right\}_{t=1}^{T_{pred}}$, a set of T_{pred} predicted positions. Then, the input is embedded into a higher D-dimensional space using a linear projection defined by the weight matrix W_x, i.e., $e_{obs}^{i,t} = x_t^{(i)\intercal} W_x$. In the same way, the D-dimension output vector is back-projected to coordinates $x_t^{(i)}$.

The transformer utilizes positional encoding to encode time for each past and future time instant t. Each input embedding $e_{obs}^{i,t}$ is time-stamped with its time t by adding a positional encoding vector p^t with the same dimension, i.e., $p^t + e_{obs}^{i,t}$.

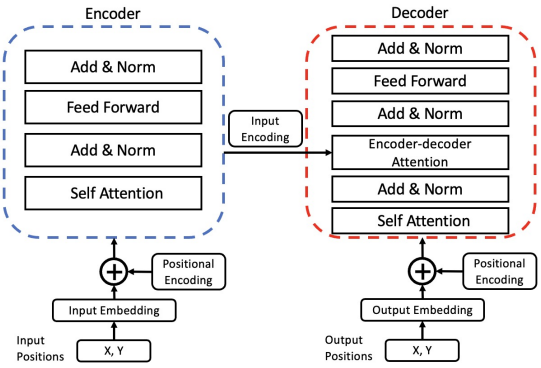

Fig. 2. Model illustration of Transformer networks

Fig. 3. Transformer prediction results

As illustrated in Fig. 2, the Transformer architecture features a modular design consisting of an encoder and a decoder, each composed of layers with three primary components: an attention module, a feed-forward fully connected module, and two residual connections that follow each of the aforementioned blocks.

The network's ability to understand complex sequences primarily comes from its attention modules. In these modules, each part of a sequence, called a "query" (Q), is compared to all other parts, called "keys" (K), using a scaled dot product. This scaling is based on the dimensionality of the query and key embeddings, denoted as d_k. The result of this comparison is then used to assign importance to the sequence parts, now referred to as "values" (V). Attention is given by $softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$. The encoding stage will create a representation for the observation sequence, and the decoder predicts the future track positions with an auto-regressive method.

The prediction outcomes are illustrated in Fig. 3, where the observations are marked with blue dots, the ground truth is represented by green dots, and the predictions are indicated by red dots.

C. TTC computation

We utilize a Time-to-Collision (TTC) method to evaluate the risk of near-crashes mathematically. We represent the trajectory of each object (i.e., vehicles and pedestrians on the road) as a set of future (x, y) coordinates for i seconds in the future, predicting the locations where the object is expected to be at various times in the future. A constant time

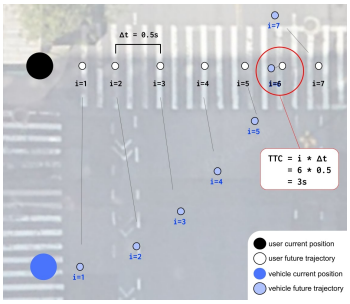

Fig. 4. Example of how Time To Collision is calculated, using visuals that are displayed in the pedestrian App

interval is held between consecutive coordinates, Δt. For the ith future coordinate, the user and other objects are represented by (U_{xi}, U_{yi}) and (V_{xi}, V_{yi}), respectively. For each pair of future coordinates, the following inequalities are measured:

$$\begin{aligned} |U_{xi} - V_{xi}| &< D_{TTC} \\ |U_{yi} - V_{yi}| &< D_{TTC} \end{aligned} \tag{1}$$

where D_{TTC} is a threshold set for how close two future coordinates can be before a predicted collision is likely to happen. If both are true, the TTC is calculated using $i' * \Delta t$, where i' is the nearest future coordinate that satisfies Eq. (1). An example is modeled in Fig. 4, the red circle illustrates the nearest set of points that satisfy Eq. (1). In this situation, if the TTC is less than a threshold T_{TTC}, a warning is issued.

It is important to note that the smaller Δt is, the more accurate the prediction will be. However, this will also come at a cost of a higher processing time to generate and compare the coordinates [16].

D. MQTT message broadcasting

MQTT is a lightweight messaging protocol widely used in the Internet of Things (IoT) for device-to-device communication. We utilize MQTT to transmit messages between the server and the user end.

Object detection and tracking results are sent as MQTT messages in JSON format. Messages include information about bounding boxes, class labels, and future trajectories. Bounding boxes are split into two classes: vehicles and pedestrians.

V. System Design

A. System architecture

We develop a phone app to display warning messages to the user for potential collisions. The phone app is connected to the COSMOS system via two intermediate servers, the COSMOS-end Server and the User-end Server, by WiFi or 5G. The COSMOS system receives the camera feed, sends an MQTT message containing vehicle and pedestrian information to the COSMOS-end Server script, which stores the information in a file ($traj.json$). This file constantly updates every MQTT message, and stores trajectories as a set of future coordinates at multiple time intervals for each vehicle and pedestrian. Simultaneously, the App on the user's phone pings the User-end Server script. Once the server script receives this ping,

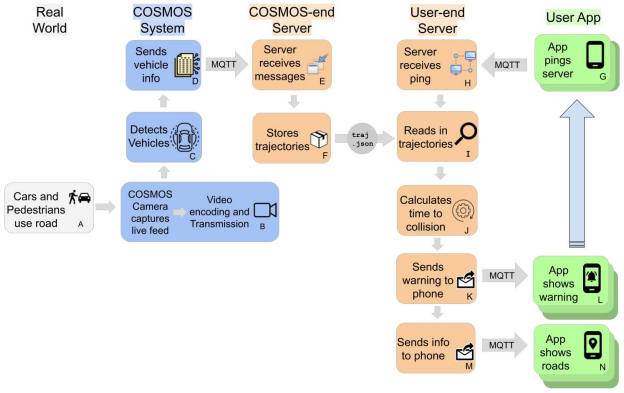

Fig. 5. Flowchart for how the entire system architecture works.

it starts to calculate the TTC for the specific pedestrian by (1) reading the trajectory for each vehicle from $traj.json$, (2) compare the user's future coordinates with every other vehicle's future coordinates, and (3) calculate the TTC to assess the risk of a dangerous collision. If a high risk (the TTC is less than the threshold T_{TTC}) is identified, issue a safety warning.

Once the TTC is calculated, or it is determined there will be no collision, a collision warning or neutral message is sent to the App, respectively. In addition, the predicted coordinates are sent to the App as well, in a different message, such that the App can construct a UI dashboard, displaying where each surrounding vehicle and pedestrian is and where the user is. In the application, a warning appears at the top of the screen, shown in red to signify danger or in green to indicate safety. Lines come out of each dot, representing the future trajectory of each vehicle for the next 4.8 seconds. Only once this process is completed will the App ping the server again to decrease the latency, which will be expanded upon later. In summary, the COSMOS system is constantly running, sending MQTT messages about the data. Both the COSMOS-end Server script and the User-end Server script are continuously running, waiting for their respective devices to send MQTT messages, and the App is continuously active on the user's phone, sending pings to the server. This process is modeled in Fig. 5.

1) Why this architecture: This architecture was designed to allow for scalability with respect to multi-device support. Only when the server receives a ping from a device, it determines the TTC for that pedestrian. If multiple devices ping the server, the server will process each request separately. In addition, the app will only ping the server after it has already received the previous warning. There is a continuous flow of warnings, going only as fast as the phone can handle, called the "ping-server loop." For phones with higher computing power or better internet, the ping-server loop will be much faster. This system has been tested to work for a range of devices.

The alternative to the current system would be sending a mass warning to every phone as soon as the COSMOS system collects new data about the road. This would reduce the latency between when the COSMOS system reads information about the road to when the warning is sent to the user. However, there are drawbacks. If messages are sent too frequently (i.e., 10 times a second) but the phone cannot handle it (i.e., it only processes it 5 times a second), there will be a backlog of messages on the phone, so warnings will be exponentially delayed. One way to overcome this is to skip frames, such that the server is only sending information every so often (i.e. every other frame). In addition to potentially missing important warnings, this doesn't fully solve the problem as it isn't scalable to other devices, can fail in areas with bad cell service, and might not allow the phone to reach its full potential.

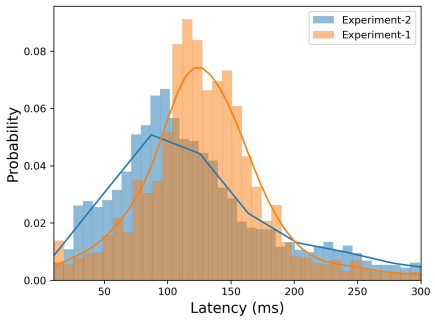

Fig. 6. Latency distribution of ping loop in field experiments.

For these reasons, the variable approach is superior in terms of scalability.

Description	Steps covered	Avg Time (s)	Avg Time (ms)
Camera feed to MQTT message generated	B-C-D	0.200	200
Camera feed to receive COSMOS MQTT message on App server	B-C-D-E	0.209	209
Camera feed to save messages on App server	B-C-D-E-F	0.211	211
Server Processing Time	H-I-J-K-M	0.009	9
Ping-Server Loop	H-I-J-K-L-M-N-G-H	0.057	57
Camera feed to Warning sent	B-C-D-E-F-H-I-J-K	0.405	405
Camera feed to Future Coords Sent	B-C-D-E-F-H-I-J-K-M	0.406	406

TABLE I
LATENCY FOR EACH STEP.

B. System deployment

1) Two field Experiments: We conducted two field experiments to test the App at the intersection of W 120th St at Amsterdam Ave in West Harlem, New York City. In this test, a pedestrian (tester) used the App on a cellphone. He was tracked by the camera in COSMOS system as they moved in the intersection, intentionally attempting actions such as jaywalking, moving closely to the trajectories of the vehicles (in a safe manner) and quickly approaching the crosswalk and stopping suddenly. These were designed to simulate a near-collision scenario between a pedestrian and vehicles without any actual risk of injury. Any warning messages would be

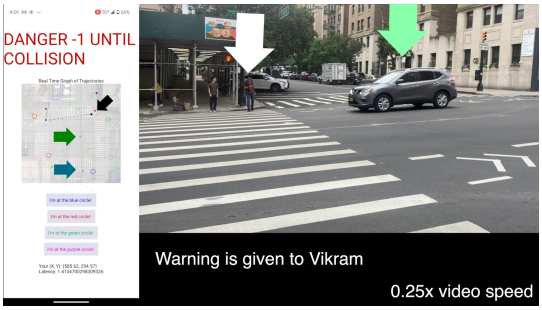

Fig. 7. Image from live field test in New York City, alongside the App interface. The arrows (overlaid onto the video, not in the App interface) shown in the App correspond to arrows over the people, marking the user (black / white arrow) and detected vehicles (blue and green arrows).

Fig. 8. Photographs from the first and second field experiments

displayed in the App, and the metrics would be stored on the backend for analysis.

We also record the ping loop latency to evaluate the system. In the first field experiment, we had 4 participants; they were guided to complete 5 rounds of experiments. During each round, participants connected to the system and walked around, generating approximately 1,000 data points of ping loop latency. In the second field experiment, 40 participants were divided into ten groups. Each group underwent a round of testing to evaluate the system on different devices and validate its scalability. The detailed parameters of the two experiments are shown in Table II. According to the distribution illustrated in Fig. 6, the ping loop latencies are stable across different devices. The mean latency of the experiment is around $135ms$ for both of the experiments.

Experiment	Date	# of participants	# of rounds	# Ping loop	Ave. latency(ms)
1st	08/20/2023	4	5	5,204	135.96
2nd	10/27/2023	40	10	10,056	138.31

TABLE II
DESCRIPTION OF TWO FIELD EXPERIMENTS

We improved the method of locating the pedestrian. Originally, the App was designed to record the pedestrian's GPS location (latitude and longitude coordinates) and send it to the server, such that it can determine the future trajectory of the pedestrian and detect collisions. However, the GPS coordinates found in the city are very unreliable due to the amount of buildings in the area. To fix this, we leverage the accuracy of the COSMOS system in detecting pedestrians. When the user opens the App, they are shown a birds-eye view of W 120 St at Amsterdam Ave intersection, with virtual circles drawn in specific places. Once they are standing in one of the colored circles, they click a button labeled "I'm standing

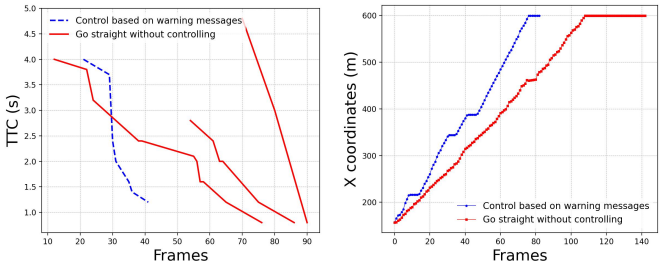

Fig. 9. TTC (left) and pedestrian coordinates (right) in CARLA simulation

on the circle." The server then finds the closest pedestrian to that circle and links the pedestrian detected in the COSMOS system to the user App connected to the server. This way, the App is accurately able to track the pedestrian.

2) Latency: In the field test, to better understand the latency of each process, we recorded the latency for certain processes and averaged them. In the "Steps covered" column, letters are utilized to describe each step that was included in the latency count. Refer to Fig. 5 for which step each letter corresponds to. The results, presented in Table I, reveal that a significant portion of the latency originates from the camera feed to the MQTT messages. The total latency from the camera to the user end is approximately 0.4 seconds. This latency has been tested and deemed sufficient for keeping pedestrians safe, discussed in the following section.

C. System Validation

In the field experiments, it is necessary to conduct testing in a completely safe manner. As a result, sometimes the user is not close enough to the vehicles to test the App. Open-source simulators are extensively utilized to validate traffic management algorithms to reduce costs in real-world experiments [17], [18]. We develop a hardware-in-the-loop methodology for assessing the App through simulation. Fig. 10 illustrates the updated diagram, which incorporates the simulator into the system.

For the specific methodology, we utilize Python API to read the vehicle information from the MQTT broker and simulate the corresponding vehicles in the digital twin world inside the CARLA simulator. We also simulate a pedestrian in the simulator as the App user. Fig. 11 illustrates the digital twin world in the CARLA simulator. In the simulator, we have the capability to manipulate the virtual pedestrian's response to the warning messages, thereby more effectively demonstrating the system's efficiency in keeping safety.

In the simulation, the virtual pedestrian is programmed to attempt crossing the road in a straight line while disregarding the traffic signal. We record the simulated coordinates and TTC while simulating. The results are illustrated in Fig. 9. Fig. 9 depicts the TTC curve for the virtual pedestrian. In this figure, the red line represents the TTC when the pedestrian continues straight, disregarding the warning, while the blue line indicates the TTC when the pedestrian stops upon receiving a warning message. We can see the TTC will significantly decrease to $0.5s$ if the pedestrian is not controlled, and the

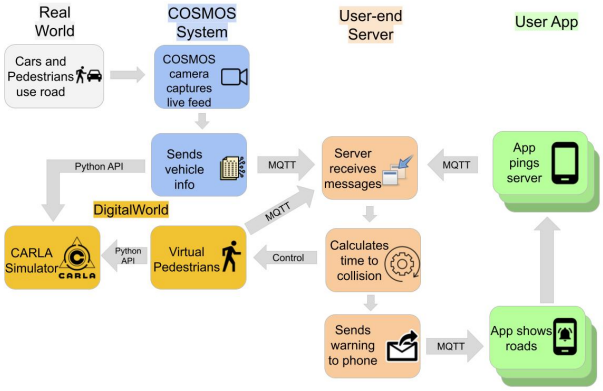

Fig. 10. Flowchart for how virtual pedestrian in CARLA simulator is involved

Fig. 11. 3D model of COSMOS testbed in CARLA

pedestrian will stop at around $1.25s$ if it can stop according to the warning. We can also have a look at the coordinates of two kinds of virtual pedestrians depicted in Fig. 9. The blue dots represent the trajectory of the controlled virtual pedestrian. This pedestrian demonstrates three instances of stopping in response to the warning messages.

In summary, the hardware-in-the-loop simulation effectively demonstrates the scalability and feasibility of the phone application, particularly in terms of its latency and accuracy.

VI. Conclusion and Open Questions

This system significantly enhances safety at individual intersections by integrating edge computing power and advanced communication technology. The effectiveness and utility of the proposed method are demonstrated through validation in a traffic simulator, employing hardware-in-the-loop simulation techniques.

One challenge is the transmission of warnings to pedestrians without cell phones. In this context, roadside devices could be instrumental. Additionally, applications implemented on the vehicle end can offer protection to VRUs. Another question pertains to the distinct surveillance, privacy, and security concerns associated with sensors like cameras. Data acquisition in the proposed system is done by geographically localized edge computing and communication nodes, which are located at individual intersections. This architecture makes it possible to design and run secure privacy-preserving algorithms and processes.

Acknowledgment

We would like to thank Sanjeev Narasimhan and Chengbo Zang for their assistance in improving the object detection and tracking module. Thank Mengxuan Liu for her assistance in generating the 3D model of the test intersection in CARLA.

References

[1] E.-H. Choi, "Crash factors in intersection-related crashes: An on-scene perspective," Tech. Rep., 2010.

[2] Z. Kostić, A. Angus, Z. Yang, Z. Duan, I. Seskar, G. Zussman, and D. Raychaudhuri, "Smart city intersections: Intelligence nodes for future metropolises," *Computer*, vol. 55, no. 12, pp. 74–85, 2022.

[3] Y. Fu and X. Di, "Federated reinforcement learning for adaptive traffic signal control: A case study in new york city," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 5738–5743.

[4] Z. Shou, X. Chen, Y. Fu, and X. Di, "Multi-agent reinforcement learning for markov routing games: A new modeling paradigm for dynamic traffic assignment," *Transportation Research Part C: Emerging Technologies*, vol. 137, p. 103560, 2022.

[5] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 3563–3576, 2018.

[6] S. Yang, E. Bailey, Z. Yang, J. Ostrometzky, G. Zussman, I. Seskar, and Z. Kostic, "Cosmos smart intersection: Edge compute and communications for bird's eye object tracking," in *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2020, pp. 1–7.

[7] M. Ghasemi, Z. Yang, M. Sun, H. Ye, Z. Xiong, J. Ghaderi, Z. Kostic, and G. Zussman, "Video-based social distancing: Evaluation in the cosmos testbed," *IEEE Internet of Things Journal*, 2023.

[8] A. Angus, Z. Duan, G. Zussman, and Z. Kostić, "Real-time video anonymization in smart city intersections," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2022, pp. 514–522.

[9] R. Bautista-Montesano, R. Galluzzi, K. Ruan, Y. Fu, and X. Di, "Autonomous navigation at unsignalized intersections: A coupled reinforcement learning and model predictive control approach," *Transportation research part C: emerging technologies*, vol. 139, p. 103662, 2022.

[10] A. Tahmasbi-Sarvestani, H. N. Mahjoub, Y. P. Fallah, E. Moradi-Pari, and O. Abuchaar, "Implementation and evaluation of a cooperative vehicle-to-pedestrian safety application," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 62–75, 2017.

[11] F. Arena, G. Pau, and A. Severino, "V2x communications applied to safety of pedestrians and vehicles," *Journal of Sensor and Actuator Networks*, vol. 9, no. 1, p. 3, 2019.

[12] S. Y. Gelbal, M. R. Cantas, B. A. Guvenc, L. Guvenc, G. Surnilla, and H. Zhang, "Mobile safety application for pedestrians," *arXiv preprint arXiv:2305.17575*, 2023.

[13] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejski, M. Sherman, Z. Kostic, X. Gu *et al.*, "Challenge: Cosmos: A city-scale programmable testbed for experimentation with advanced wireless," in *Proceedings of the 26th annual international conference on mobile computing and networking*, 2020, pp. 1–13.

[14] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *European Conference on Computer Vision*. Springer, 2022, pp. 1–21.

[15] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th international conference on pattern recognition (ICPR)*. IEEE, 2021, pp. 10 335–10 342.

[16] D. Greene, J. Liu, J. Reich, Y. Hirokawa, A. Shinagawa, H. Ito, and T. Mikami, "An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 942–953, 2011.

[17] S. Liu, Y. Wang, X. Chen, Y. Fu, and X. Di, "Smart-eflo: An integrated sumo-gym framework for multi-agent reinforcement learning in electric fleet management problem," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 3026–3031.

[18] Z. Mo, W. Li, Y. Fu, K. Ruan, and X. Di, "Cvlight: Decentralized learning for adaptive traffic signal control with connected vehicles," *Transportation research part C: emerging technologies*, vol. 141, p. 103728, 2022.