2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Real-time Hazard Prediction in Connected Autonomous Vehicles: A Digital Twin Approach

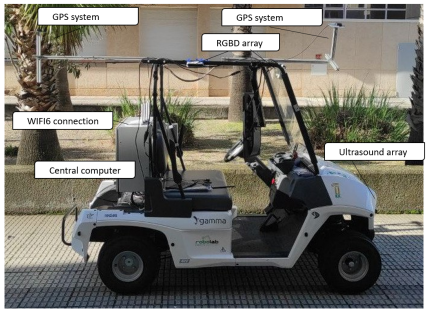
Sergio Barroso, Noé Zapata, Gerardo Pérez[1], Pablo Bustos and Pedro Núñez

***Abstract*— The growing interest in connected autonomous vehicles (CAVs) has intensified the focus on technologies and algorithms that enhance behavior, comfort, and safety. Among these, the concept of Digital Twins (DT) represents an emerging field of research that is now beginning to be applied to autonomous systems. Traditional Advanced Driver-Assistance Systems (ADAS) can prevent real-time collisions using sensor data. However, we propose that employing a DT can enable the accounting for complex, simulated decisions before they occur in reality. This paper introduces an initial model of a Digital Twin, founded on an internal simulator aligned with vehicle control architecture, for real-time hazard prediction and effective decision-making. Our DT synchronizes with the vehicle's state to simulate various hazardous scenarios in advance, allowing for preemptive actions. To support our hypothesis, we introduce an algorithm for the early detection of potential collisions between CAVs and pedestrians through the unsupervised simulation of diverse traffic scenarios. This solution integrates the CORTEX cognitive architecture with CARLA for internal simulation, leveraging probabilistic models to select optimal scenarios. Employing data from external pedestrian cameras, a particle filter predicts the most probable pedestrian trajectories via DT simulations, thereby informing safe maneuvers. Although the algorithm itself is established, the novelty of our approach lies in incorporating a simulator within the digital twin. This simulator, informed by real-time data on the vehicle's and environment's state, facilitates appropriate responses to unpredictable behaviors. We have conducted extensive tests with an actual autonomous electric vehicle on a university campus to validate the system's predictive and adaptive functions.**

I. Introduction

Digital Twin (DT) technology is emerging as a key tool for optimizing complex systems like Connected Autonomous Vehicles (CAVs) [1]. DTs, virtual replicas of physical systems, have gained accuracy and power thanks to advancements in artificial intelligence, IoT, Cyber-Physical Systems, and cloud computing [2].

In contrast to traditional Advanced Driver-Assistance Systems (ADAS), which are primarily reactive and focus on collision avoidance, DTs offer predictive capabilities that can enhance decision-making and safety. By utilizing vehicle sensor data, DTs simulate various driving scenarios, serving as a comprehensive, anticipatory monitoring tool. Unlike ADAS, which struggles with complex pedestrian interactions, DTs enable real-time simulation of multiple future scenarios for nuanced decision-making. The true value of DTs lies in building mathematical models from this real-time data, allowing interrogation to understand future vehicle behavior and generate autonomous commands. This contributes to more efficient, reliable mobility, bolstered by the substantial live data that can be fed into the model [2].



(a)

(b)

Fig. 1: 1a shows Melex golf car, which will be used to test a digital model with its sensor. Fig. 1b shows both the real and simulated world.

In Figure 1a, a CAV with multiple sensors connects to a central computer, which sends data to the cloud. Figure 1b shows the DT model of the vehicle and environment. Advanced physics engines and rendering simulate vehicle behaviors like sensor activity and command outcomes, aiding quick, informed decision-making.

This paper offers several contributions: (i) it extends an existing control architecture, CORTEX [3], to incorporate an internal simulator, essentially converting it into a DT based on sensor data and external devices. This is integrated with the realistic CARLA simulation environment [4]; (ii) we introduce an algorithm enabling the DT to run multiple real-time simulations, comparing results to actual measurements for Particle Filter (PF) estimation of pedestrian trajectories. While the PF algorithm is well known, it is just one example of integration and the strength of a digital twin to detect

[1]All authors are with RoboLab research group, University of Extremadura, Caceres, Spain pnuntru@unex.es

risky situations using multiple simulations; and (iii) the architecture is validated in real scenarios, where we evaluate the use of the digital twin with the proposed algorithm to make decisions affecting vehicle speed based on these simulations.

II. Related work

Autonomous vehicles operate without human intervention using sensors, cameras, radar, LIDAR, and AI algorithms [5]. CAVs aim to improve road safety, alleviate traffic, enhance mobility, and lower environmental impact. Yet, they face technical challenges such as reliability and pedestrian safety [6]

DTs are one of the emerging technologies that can address some of these challenges. DTs can enable various applications, such as simulation, optimization, prediction, diagnosis, control, and maintenance of physical systems [7]. Moreover, DTs can facilitate communication and collaboration between stakeholders involved in the design, development, and operation of physical systems [8].

In CAV, DT can support different aspects of driver assistance, such as perception, planning, decision-making, and control. For example, in [9], the authors review the state-of-the-art applications of DT technology in various aspects of intelligent electric vehicles, such as predictive mobility, advanced driver assistance systems, vehicle health monitoring, battery management systems, power electronic converters, and power drive systems. In [10], the authors describe real-world experiences with DTs, where they develop a paradigm for a practical autonomous driving system. In the context of driving assistance, a DT can capture and simulate various aspects of pedestrian safety, such as sensor measurements, vehicle dynamics, traffic conditions, and driver behavior [11], [12]. A DT can also enable data-driven optimization, verification and validation, fault diagnosis and recovery, and human-in-the-loop learning for driver assistance systems [13], [14]. This article investigates DT for driver assistance, focusing on pedestrian safety. We provide a DT architecture that allows for the simulation of multiple situations in real-time and use a particle filter-based methodology to maintain a representation of the virtual replicas. Tracking pedestrians using the PF makes acting on the vehicle's braking system possible, improving vehicle safety.

III. An inner simulation architecture for connected vehicles.

Figure 2 provides an overview of the proposed DT framework for autonomous driving of connected vehicles. The figure shows three blocks. The block on the left represents the physical world, while the middle block shows the CORTEX cognitive architecture. The shared Working Memory (WM) is depicted as a graph whose nodes represent (among other things) grounded elements of the physical world, and the edges represent relationships among them. A more detailed description of CORTEX and how agents communicate its information can be found in [3]. Finally, the rightmost block represents the internal simulator or DT of the vehicle, which has been integrated into CORTEX as a drivable module that can be synchronized, started, or queried at any time.

A. The Physical World

The environment, vehicle with its devices, and external sensors communicating through Vehicle-2-everything (V2X) protocols make up the physical world. In this work, an electric car has been equipped with a rig of RGBD cameras mounted around the roof to provide a 360^o field of view. Additionally, the lower part of the external plastic cover has an array of interleaved ultrasound and point LIDAR sensors. The vehicle's global position is determined using a GPS-RTK system. The vehicle communicates with a remote monitoring computer via a WiFi6 router.

B. Cortex Architecture

The cognitive architecture CORTEX serves as the controller for the CAV, and a schematic view of it can be seen in Figure 3. The most important aspect of the experiment is the WM, which is located in the center. This WM is a distributed, shared graph with M nodes and E edges, each having its associated attribute list. The DT stores all the physical and semantic information of the environment.

Since the graph is distributed ($i.e.$, agents may run in different machines), it exists as a set of local copies maintained by the agents involved. These copies are synchronized using Conflict-Free Replicated Data Type (CRDT) [15] and Data Distribution Service (DDS) [16] technologies, which allows consistent in the system. Agents ($e.g.$, monitor, environment perception, or navigation software agents) update the graph by adding new sensor readings, estimated geometric relations, or logical predicates between nodes. The car and all its sensors and actuators are nodes in the WM. Each device is connected to the vehicle through a specific type of edge called RT, which encodes its physical position and orientation. The primary purpose of the WM is to provide an updated context for all agents to make better-informed decisions. Figure 3 provides a representative example of the navigation process, featuring the vehicle along with its onboard sensors, an external camera, and any identified pedestrians. Presented as a graph, this representation is accessible to all agents in the architecture, each maintaining a synchronized copy.

C. Carla Simulator

The third component in Figure 2 is a built-in simulator managed by the CORTEX architecture. We opted to use the CARLA simulator due to its wide range of tools for training and validating autonomous driving systems. To create the 3D representation of our working environment (as shown in Figure 1b), we employed RoadRunner, an interactive editor from MathWorks specializing in designing 3D traffic routes. Additionally, we used Blender to create the 3D model of our autonomous vehicle. Using CARLA, which provides realistic assets such as buildings, pedestrians, and cars, along with the 3D models of our test environment and vehicle, we could recreate traffic scenarios based on specific events

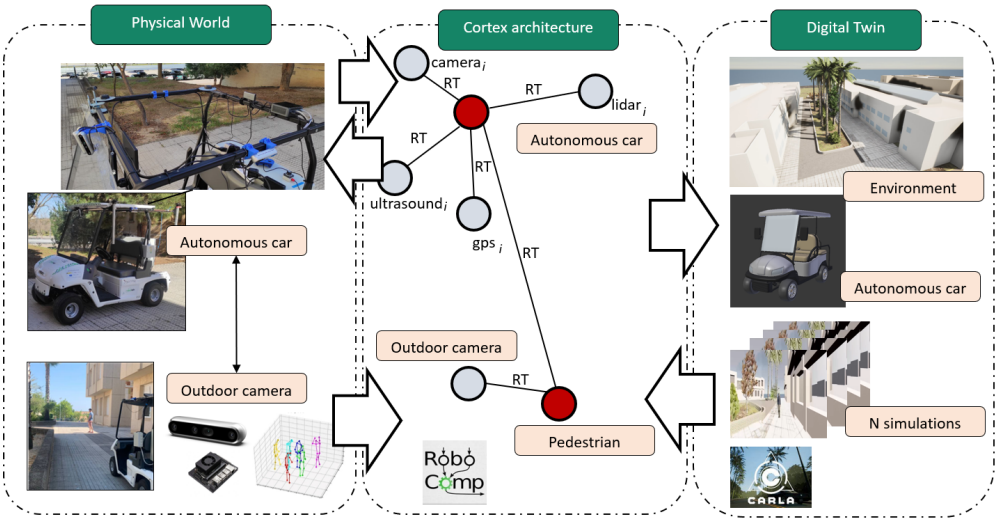


Fig. 2: Overview of the DT architecture

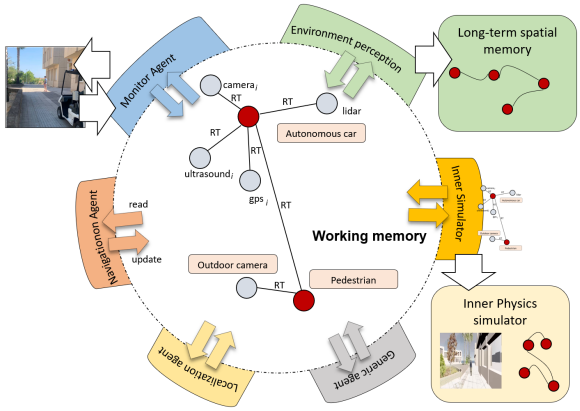


Fig. 3: Schematic view of the current version of CORTEX.

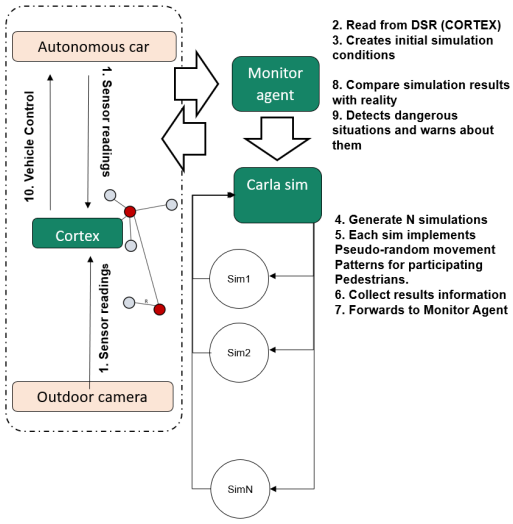


Fig. 4: Overview of the algorithm for enhancing the safety of pedestrians using DT.

in our real environment. One of the main accomplishments of this research has been the incorporation of CARLA as a controllable resource for the control architecture. The agent that controls CARLA can work in two modes: a) by synchronizing the simulator with the state of the WM and b) by initiating a set of simulations from the current state and with different parameters until a final condition is met or during a maximum time. In our case, the simulations are meant to explore potential courses of action of the detected pedestrian in the current scenario and prepare the car for a quick reaction; however, we can use these simulations for different behaviors.

IV. Algorithm for Enhancing the Safety of Pedestrians Using DT

Within this deployment of the CORTEX architecture, we present an algorithm to derive an informed control action in risk situations created by the presence of pedestrians. The process is illustrated in Figure 4. It involves gathering information in advance from both the autonomous vehicles, which sends data collected by its sensors to the CORTEX shared memory, and from pedestrians in the vehicle's surroundings, whose position in the world is also transmitted via V2X communication and shared in CORTEX (1). Several software agents with different responsibilities access CORTEX, including the **Monitor agent** (2). This agent is in charge of producing a series of initial simulation conditions, such as the simulation identifier, simulation duration, and list of actors present in the work environment, along with their position in it and their role, which it then forwards to the CARLA SIM agent (3).

The **CARLA Sim** agent communicates directly with the CARLA simulator and generates a set of N simulations based on the initial conditions received from the Monitor agent (4). These simulations run in parallel at a much faster speed than real-time and for each of them, the behavior of the actors involved is defined by pseudo-random behavior

(5). Our algorithm's objective is to filter out only those simulations that match the reality perceived by the vehicle. To achieve this, we use a particle filter. The PF is a popular method for estimating the state of nonlinear systems with a partially observed state based on the Rao-Blackwell formula (see Casella and Robert's work [17]). Each generated simulation considers the randomness of the trajectories of the different actors and their impact on driving safety. With this information captured in advance, we can act on the vehicle's braking system to improve pedestrian safety (6-10).

In this paper, the set of particles represents a hypothesis of pedestrian (and can be extended to include vehicles' poses) trajectories that we consider correct (i.e., Maximum a Posteriori estimation). Then, using this assumption, every particle is associated with a simulation of our DT and will maintain its trajectories. Let x_t be the state space defined by the main parameters of the simulation, including the position and velocity of each pedestrian in the scene for the set of N simulations. Let $x_t^{(i)_n}$ be the state of pedestrian i at time t in the simulation n, where $i = 1, \ldots, M$, $n = 1, \ldots, N$, and t is the current time step.

$$x_t^{(i)_n} = \begin{bmatrix} p_t^{(i)_n} \\ v_t^{(i)_n} \end{bmatrix} \tag{1}$$

For each simulation n, $p_t^{(i)_n}$ represents the position of person i, and $v_t^{(i)_n}$ represents their velocity at time t. Additionally, let $y_t^{(i)}$ be the measurement of person i (i.e., pedestrian position) at time t from sensor readings, which is stored in the WM.

$$y_t^i = p_t^{(i)_n} + \epsilon_t^{(i)} \tag{2}$$

where $\epsilon_t^{(i)}$ is measurement noise, which depends on the real sensor. A similar notation could be used to include the vehicle's pose estimate. Each particle is a sample from the posterior distribution and is represented by the tuple $(x_{1:t}, w_t)$, where w_t is the weight associated with the particle at time t. The weights approximate the posterior distribution, with particles with higher weights having a more significant influence on the approximation. Therefore, the PF performs the following steps:

1) Initialization: At time $t = 1$, create N particles $(x_1^{(n)}, w_1^{(n)})n = 1^N$ from an initial distribution. In our approach, this initial distribution for the pedestrian pose is normal, with an average equal to the pedestrians' pose stored in the WM and a standard deviation of σ_p. The speed will also be derived from a normal distribution with an average of $1.5m/s$ (normal speed of a random person) and a standard deviation of σ_v. We set $w_1^{(n)} = 1/N$.
2) Importance Sampling: For each time step $t = 2, 3, \ldots, T$, do the following steps:
   - For each simulation $(x_{1:t-1}^{(n)}, w_{t-1}^{(n)})$, create a new state $x_t^{(n)}$ from the state transition distribution $p(x_t \mid x_{t-1}^{(n)})$. In our approach, this transition distribution considers the results of each realistic scene simulation by different conditions (pedestrians and vehicles).
   - Compute the importance weights $w_t^{(n)} \propto p(y_t \mid x_t^{(n)})$ for each particle. Note that the weights are proportional to the likelihood of the observation given the particle's state. At this point, we use the updated information from the physical world stored in the WM. The final pedestrian poses for each actor in each simulation are compared with the actual pose of each actor. For a pose to be considered correct, all actors must have an error in the simulated final pose compared to the real pose below a certain threshold.
   - Normalize the weights so that $\sum_{i=1}^{N} w_t^{(i)} = 1$.
   - Resample particles with replacement from the set $\left\{(x_{1:t}^{(n)}, w_t^{(n)})\right\} n = 1 \ldots N$, with probability proportional to their weights. This step ensures that particles with higher weights are copied more often, resulting in a new set of N simulations $\left\{(x1 : t^{(n)}, w_t^{(n)})\right\}_{n=1\ldots N}$. The systematic resampling method is chosen among the multiple algorithms to perform resampling.
3) Filtering: after resampling, the particle set represents the current posterior distribution. The estimated state of the system can be computed as a weighted average of the particles:

$$\hat{x_t} = \sum_{n=1}^{N} w_t^{(n)} x_t^{(n)} \tag{3}$$

where $\hat{x_t}$ is the estimated state of the system at time t. The estimated trajectory of the pedestrian can then be obtained by tracking the estimated position of the pedestrian over time.

After the simulations are complete, based on the simulation time set in the initial conditions, the CARLA Sim agent collects the results and sends them back to the Monitor agent for processing. These results include various parameters for each simulation, such as i) collision, which indicates whether the autonomous vehicle has collided with any actor and, if so, which actor it hit with; and ii) a list of actors and their positions at different points in the simulation.

After filtering out the invalid simulations, processing the correct results provides information to the WM in CORTEX. This information is represented as a **Virtual Collision edge** that connects the vehicle node to the actor(s) with which it collided during the simulation. This edge has two attributes: "Collision", which represents the percentage of simulations where the vehicle has collided with the actor, and "Time to Collision", which is a list that stores the time to collision for each simulation.

Using these edges and their information, the agent responsible for controlling the vehicle can adjust its behavior ($e.g.$, car velocity) based on the collision probabilities and the time to collision.

V. Experimental results

Tests were conducted to assess the proposed system's performance and ability to predict pedestrians' behavior along an autonomous vehicle's route in real-time, thus detecting possible dangerous situations. Specifically, in these tests, a CAV travels along a path in the Polytechnic School of Cáceres while a pedestrian moves through the same area. The Deep Neural Network (DNN), known as You Only Look Once (YOLO)[1][18] effectively identifies all pedestrians, and their skeleton is subsequently analyzed by a second DNN, JointBDOE[2], which estimates their orientation [19]. Additionally, the ByteTrack algorithm [20] is employed to track the pedestrians, providing them with a consistent identification tag.

A. Validation of the Digital Twin for the use case

The results are presented using the parameters obtained from the analysis of the simulations performed with the DT, and the speed of the autonomous vehicle is calculated based on these parameters. The following parameters are used:

- Collision probability: The percentage of valid simulations in which a collision with another actor occurs.
- Time remaining until the next collision in seconds.
- Distance between vehicle and pedestrian in meters.
- Vehicle speed in meters per second.

The tests were conducted as follows (see Fig. 5): (i) the CAV receives information from all sensors, updating the WM, (ii) based on this information, n = 8 simulations of 6s are generated in the DT, modifying values related to the pedestrian's behavior pattern over time; (iii) the simulations are validated against real information acquired by the sensors, removing erroneous simulations and generating new ones; (iv) the behavior of the vehicle is modulated based on the simulation results; (v) New initial simulation conditions are generated by updating the variables related to pedestrian movement and actor positions based on previous simulation results and sensor data.

An example of the simulation results obtained can be seen in Fig. 6. The pedestrian (represented by the red dot) remains static outside the path boundaries, followed by the CAV (violet dot). A yellow dot represents each final position of the pedestrian in those valid simulations, while those of the CAV are depicted in green. The result shown in Fig. 6 indicates a high degree of uncertainty in the direction the pedestrian can move.

In contrast, Fig. 7 shows the result of a pedestrian moving in a given direction, causing a correction in the simulated behavior of the pedestrian and generating simulations closer to the direction the pedestrian is moving in. Throughout the simulations, the existence of collisions or dangerous approaches increases the probability of collision and records the remaining time (in simulation) until the crash occurs. Subsequently, the agent monitoring the simulation collects this information and influences the vehicle's behavior to avoid collision in the real world. Additionally and iteratively, it generates a new set of simulations.

In the upper graph on Figure 8, it can be seen how as collisions start to occur (increase in the probability of collision), the behavior of the real vehicle is modulated by reducing the speed and even stopping the car, moving to a safer navigation mode. Each of the collisions detected in the simulation is recorded, helping to generate an estimate of the time to collision.

The anticipation distance depends on the vehicle's and the pedestrian's speed. As can be seen in Fig 8, once the car has stopped (t=11), it must remain in this state until the probability of collision (pedestrian off the road) disappears, regaining mobility once the pedestrian is out of danger.

B. Real-time performance of the algorithm

Fast and reliable decision-making is critical to validate the DT system's usability. This section assesses the digital twin's simulation execution times, which depend on various factors: server number, simulations per thread, and the real-time duration for simulation. For this experiment, we simplified the previous case and set the vehicle straight towards the pedestrian. Figure 9 shows the average execution times relative to the real simulation time and the number of simulations per CARLA instance. Notably, longer simulated times and more simulations per instance increase execution times. Our system demands swift responses, so we dismiss any parameter sets exceeding one-second execution and view those above 0.6 seconds as unfavorable. As indicated in the previous section, with 8 simulations and with a time for each simulation 6s, we obtain adequate execution times.

Additionally, we highlight our approach's scalability. In more complex scenarios, with multiple pedestrians or vehicles, we can increase the number of simulations by increasing the number of instances to the CARLA server, running in parallel. Computing times would not be significantly affected. This makes our approach scalable and efficient, even in areas with high pedestrian activity.

[1]We use YoloV8 from Ultralytics. https://www.ultralytics.com/

[2]https://github.com/hnuzhy/jointbdoe

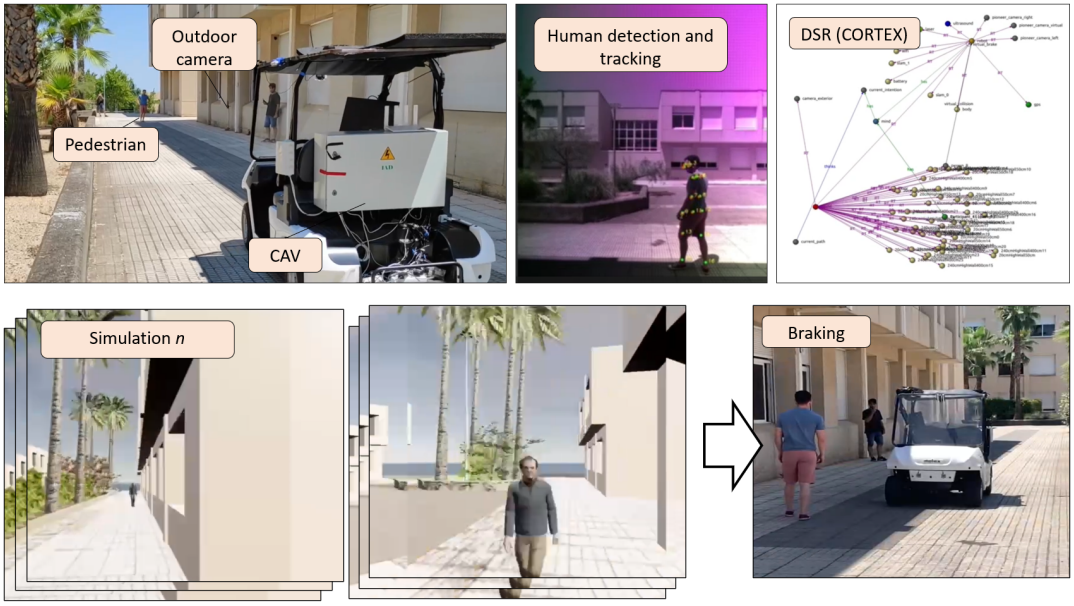


Fig. 5: Various snapshots of the experiments are depicted. The top row, from left to right, presents the experimental setup, the pedestrian detection algorithm, and the status of the WM. The bottom row, also from left to right, displays different states of one of the simulations and the outcome (brake) produced by the system after implementing the PF.

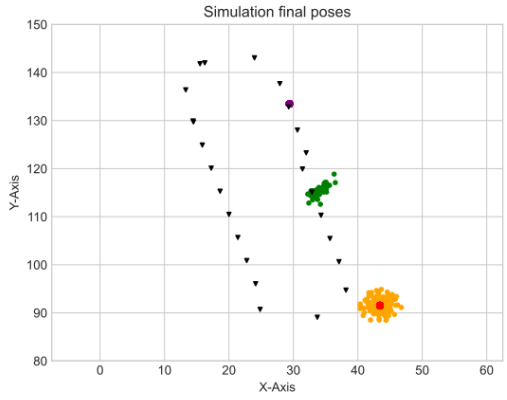


Fig. 6: Pedestrian (red dot) stationary outside the path (black marks) of the CAv (violet dot). Simulation results are orange (pedestrian) and green (CAV), respectively.

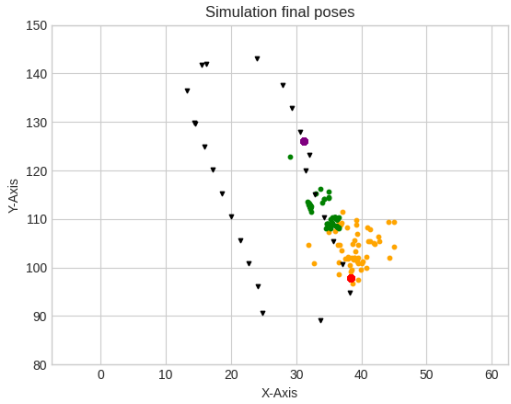


Fig. 7: Pedestrian (red dot) moving within the path (black marks) of the CAV (violet dot). Simulation results are orange (pedestrian) and green (CAV), respectively.

VI. Conclusions

This paper presents a new method to detect potential collisions between connected autonomous vehicles and pedestrians. Our approach uses the Digital Twin scheme, incorporating real-world data into a comprehensive internal simulation framework. We have integrated the CORTEX cognitive architecture with the CARLA simulation environment to create a robust Digital Twin model. This fusion facilitates concurrent real-time simulations of various traffic scenarios, enabling predictive risk assessment through probabilistic modeling and the formulation of preventive strategies. The cornerstone of the probabilistic algorithm we present is the well-known particle filter. However, our work's true innovation is encapsulated in our digital twin's capabilities, designed to anticipate and react to potential hazards before they manifest themselves, marking a significant advance in proactive safety measures in CAV operations.

We tested our system on a real CAV on the university campus to validate it. The results show how our solution successfully detects and avoids collisions with pedestrians in various scenarios, anticipating their possible trajectories and potential interactions with the vehicle. We think anticipation using an embedded Physics simulator is becoming a key player in robotic cognitive architectures. However, much work remains to be done on integrating and controlling this powerful asset in current autonomous systems.

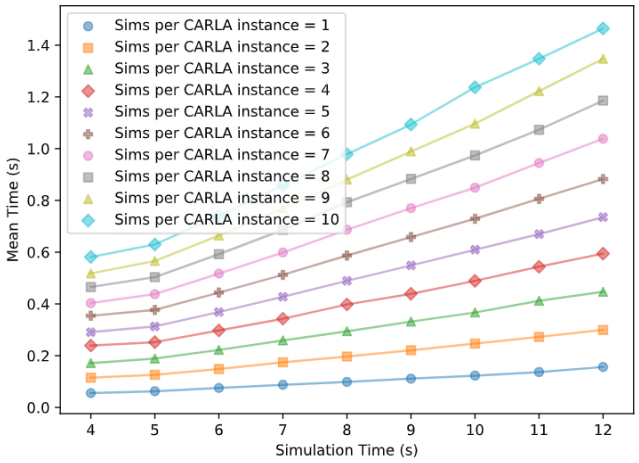


Fig. 9: Average execution time of the simulations depending on the number of simulations per CARLA instance and the total time to simulate in each simulation.

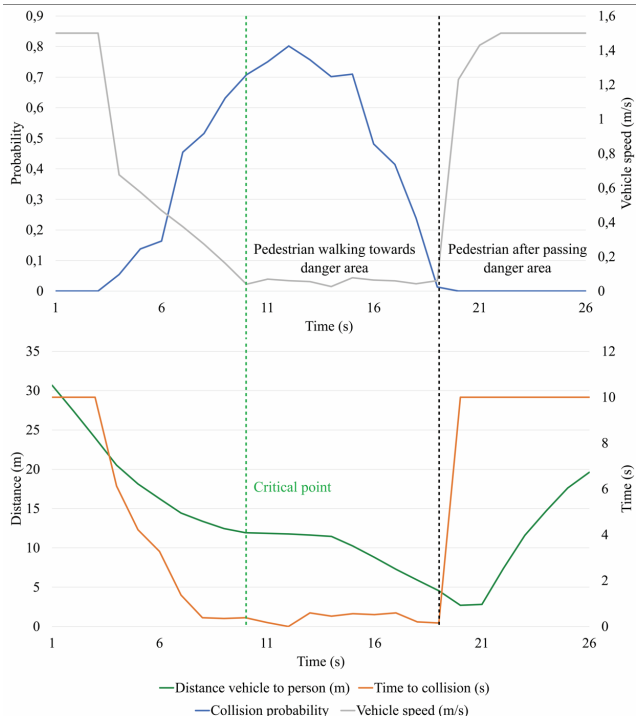


Fig. 8: Evolution of the relevant parameters during the experiment.

ACKNOWLEDGMENT

This work has been partially funded by TED2021-131739-C22, supported by Spanish MCIN/AEI/10.13039/501100011033 and the European Union's NextGenerationEU/PRTR, by the Spanish Ministry of Science and Innovation PDC2022-133597-C41 and by FEDER Project 0124 EUROAGE MAS 4 E (2021-2027 POCTEP Program)

REFERENCES

[1] Grieves, Michael. (2015). Digital Twin: Manufacturing Excellence through Virtual Factory Replication. Global Journal of engineering science and researches, 2014.

[2] Madni, A and Carla, C. and Scott, L, "Leveraging Digital Twin Technology in Model-Based Systems Engineering", in Systems, vol. 7, num. 1, pp. 2079–8954, MDPI, 2019.

[3] García, JC "G: a low-latency, shared-graph for robotics cognitive architectures", Master Thesis. Escuela Politécnica, University of Extremadura. 2021.

[4] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V., "CARLA: An open urban driving simulator." In Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16, 2017.

[5] Kockelman, KM and Avery, P and Bansal, P and Barrett, WJ and Boesch, PM and Chen, DZ and Dresner, K and Gadda, S and Gao YN, "Autonomous vehicles: Opportunities challenges and future implications for transportation policies", in Journal of Public Transportation, Springer, vol. 20, num. 1, pp 3–24, 2017.

[6] Schoettle, B and Sivak, M, "A survey of public opinion about connected vehicles in the U.S., the U.K., and Australia", in 2014 International Conference on Connected Vehicles and Expo (ICCVE), 2014, pp. 687–692

[7] Tao, F and Cheng, J and Qi Q and Zhang M and Zhang H and Sui F, "Digital twin-driven product design framework", in International Journal of Production Research, vol. 56, num. 12, pp. 3935–3953, 2018.

[8] Luthra, S and Garg, D and Haleem, A and Mangla, SK and Jakhar SK, "Digital twin: an emerging disruptive technology for Industry 4.0—current trends issues applications domains opportunities challenges—a review", in The International Journal of Advanced Manufacturing Technology, vol 108, num. 11, 2020.

[9] Alshammari, A and Alghamdi, A and Alghamdi, M and Alzahrani, M and Alharbi, A and Alhazmi, A and Alharbi, A and Almalki, A, "Towards the future of smart electric vehicles: Digital twin technology for predictive mobility and vehicle health monitoring", in Renewable and Sustainable Energy Reviews, Vol. 146, Elsevier, 2021.

[10] Yu, B and Chen, C and Tang, J and Liu, S¡ and Gaudiot JL, "Autonomous Vehicles Digital Twin: A Practical Paradigm for Autonomous Driving System Development," in Computer, vol. 55, no. 9, pp. 26-34, Sept. 2022.

[11] Almeaibed, S and Al-Rubaye, S and Tsourdos, A and Avdelidis, NP, "Digital Twin Analysis to Promote Safety and Security in Autonomous Vehicles" in IEEE Communications Standards Magazine, vol. 5, no. 1, pp. 40-46, 2021.

[12] She, Y and Zhang, J, "Cognitive Digital Twin for Driving Assistance", in https://sheyining.github.io/projects/cognitive_model/, accessed on 27 Feb 2023.

[13] Allamaa, JP and Patrinos P Herman, T and Duy Son, T, "Sim2real for Autonomous Vehicle Control using Executable Digital Twin", in 10th IFAC Symposium on Advances in Automotive Control AAC 2022.

[14] Zhang, N and Bahsoon, R and Tziritas, N and Theodoropoulos, G, "Explainable Human-in-the-loop Dynamic Data-Driven Digital Twins", in https://arxiv.org/abs/2207.09106, accessed on 27 Feb 2023.

[15] Shapiro, M and Preguiça, N and Baquero, C and Zawirski, M, "Conflict-Free Replicated Data Types", in Défago, X., Petit, F., Villain, V. (eds) Stabilization, Safety, and Security of Distributed Systems. SSS 2011. Lecture Notes in Computer Science, vol 6976. Springer, Berlin, Heidelberg.

[16] eProsima, "FastDDS" https://github.com/eProsima/Fast-DDS.

[17] Casella, G. and Robert, C. P. "Rao-Blackwellisation of sampling schemes", in Biometrika 83, num. 1, pp. 1–94.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[19] H. Zhou, F. Jiang, J. Si, and H. Lu, "Joint Multi-Person Body Detection and Orientation Estimation via One Unified Embedding," arXiv preprint arXiv:2210.15586, Mar. 2023. [Online]. Available: http://arxiv.org

[20] 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, 2022, pp. 1–21.