

Received 17 June 2026, accepted 13 July 2026, date of publication 16 July 2026, date of current version 23 July 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3714040

RESEARCH ARTICLE

A Hybrid Automaton-Based Vehicle Digital Twin for Verifiable Pedestrian Protection

LORENZO BACCHIANI¹, ANDREA BEDEI¹, (Graduate Student Member, IEEE), ÖZLEM KAYA¹, ROBERTA PRESTA², MARIO BRAVETTI¹, AND ROBERTO GIRAU¹, (Senior Member, IEEE)

¹University of Bologna, 40126 Bologna, Italy

²Centro Scienza Nuova, Suor Orsola Benincasa, 80135 Naples, Italy

Corresponding author: Lorenzo Bacchiani (lorenzo.bacchiani@unibo.it)

This work was supported in part by the MOBilità Sostenibile Trasporti (MOST)-Sustainable Mobility National Research Center; in part by European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR)-MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4-D.D. 1033 17/06/2022) under Grant CN00000023; in part by European Union's Horizon Europe Research and Innovation Program Chips Joint Undertaking (JU) DistriMuSe Project through the Distributed Multi-Sensor Systems for Human Safety and Health under Grant 101139769; and in part by EU and National Funding Agencies of the participating countries.

ABSTRACT Autonomous vehicles must make reliable safety decisions despite perception noise and tight timing constraints. This work presents a Vehicle Digital Twin (VDT) architecture that incorporates a formally verified hybrid automaton as its supervisory controller. The automaton specifies discrete safety modes, continuous timing variables, and perception-driven transitions, and is verified offline using SMT-based LTL model checking to ensure determinism, freedom from deadlock, and bounded reaction time. At runtime, buffered perception inputs and staleness checks are used to trigger only those transitions consistent with the verified model, maintaining robustness to noisy or transient detections. We instantiate the approach for pedestrian-protection and evaluate it on standardized Euro NCAP scenarios within CARLA. The system consistently produces stable and interpretable braking decisions and maintains reaction times within the verified bounds, including under perception uncertainty. These results indicate that integrating a verified hybrid automaton directly into the VDT's operational loop is a practical route toward runtime-assured safety supervision in perception-driven vehicle systems.

INDEX TERMS Pedestrian protection, hybrid automata, vehicle digital twin, formal verification, safety-critical systems.

I. INTRODUCTION

The complexity of Cyber-Physical Systems (CPSs), particularly in safety-critical domains such as Advanced Driver-Assistance Systems (ADAS) and autonomous vehicles, imposes stringent requirements for ensuring predictable and verifiable behavior in uncertain environments [1]. Traditional validation approaches, e.g., simulation-driven scenario testing, provide empirical confidence but cannot guarantee that safety properties hold across all admissible operating conditions [2], [3]. This limitation is well recognized in functional-safety standards, e.g., ISO 26262 [4], which emphasize the need for design-time assurances of determinism, bounded reaction time, and freedom from unintended behavior [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Vignati.

Two complementary paradigms have emerged to address different aspects of this challenge. Formal methods, e.g., Hybrid Automata (HA) [6], offer mathematically rigorous abstractions capable of capturing the interplay between discrete control logic and continuous vehicle dynamics. Through invariants, guards, and reachability analysis, they enable reasoning about safety conditions such as braking consistency, mode determinism, and constraint satisfaction. Conversely, Digital Twins (DTs) [7], [8] provide a real-time synchronization framework in which a virtual replica continuously mirrors the state of the physical vehicle. Modern Vehicle DTs (VDTs) [9], [10] support high-frequency data ingestion, prediction, and adaptive control, enabling the system to react to perception updates, diagnose anomalies, and forecast future states.

While both paradigms have matured significantly, their integration remains limited in the context of runtime

safety assurance. Existing HA-based verification methods typically operate offline and assume idealized inputs, making them difficult to apply in realistic perception-driven environments. At the same time, most VDT implementations focus on prediction, or lifecycle monitoring rather than providing deterministic, formally enforced safety guarantees at runtime. As reported in [11], only a small subset of research studies explore the coupling of formal verification with operational DTs, and the existing ones generally target design-time analysis rather than real-time supervisory control. This leaves an architectural gap: current VDTs seldom include a verified decision layer capable of enforcing safe behavior during operation.

This paper introduces the Hybrid Automaton-Based Vehicle Digital Twin (HA-VDT) to address this gap by embedding a formally verified HA within the VDT operational core. In this architecture, the HA acts as a real-time supervisory controller whose discrete states and transitions have been expressed using First-Order Logic (FOL), Linear Temporal Logic (LTL) and verified via SMT-based reasoning [12]. Perception-derived quantities are continuously mapped onto the HA state space, so that data-driven decisions follow pre-verified safety constraints within the discrete model. This transforms the VDT from a purely predictive or descriptive digital model into a runtime safety supervisor with explicit, interpretable control logic based on formally verified discrete transitions. To demonstrate the practicality of the HA-VDT architecture, we instantiate it for a representative safety-critical function: pedestrian protection. The evaluation, performed using the CARLA¹ simulator, is designed to validate that the HA-VDT architecture maintains its formally verified guarantees under realistic, perception-noisy conditions. Our goal is not to provide an empirical performance benchmark, but to demonstrate that runtime behavior, including determinism, bounded reaction time, and robust consensus logic, is consistent with the properties established by formal verification. We leverage standardized Automatic Emergency Braking (AEB) Pedestrian Euro NCAP [13] scenarios as an industry-accepted benchmark to corroborate the architecture practical feasibility as a verifiable layer for safety-critical automotive functions. Hence, the novel contributions of this work are:

- The unified HA-VDT that embeds a formally verified HA directly into the real-time operational core of the VDT, transforming it from a purely predictive/descriptive model into an active runtime safety supervisor.
- Design and formal specification of a HA tailored to pedestrian protection as well as safety and liveness properties in FOL and LTL.
- Encoding of the automaton and its properties in the Z3 constraint solver, followed by formal verification.
- Empirical demonstration in the CARLA simulator on standardized Euro NCAP pedestrian scenarios, showing that the HA-VDT sustains stable, interpretable braking

behavior, bounded response times, and audit-ready decision traces while preserving all formally verified properties at runtime.

These contributions provide a practical and verifiable approach to embedding formally proven discrete supervisory logic directly into the operational loop of a perception-driven vehicle digital twin, offering a promising step toward runtime-assured, interpretable, and auditable safety in autonomous driving systems.

The complete code for reproducing the experiments presented throughout this work, the Z3 encoding and formalization of the automaton and properties are available at [14].

The remainder of this paper is structured as follows. Section II reviews background on hybrid automata and VDTs. Section III analyzes the state of the art and formalizes the integration gap addressed in this work. Section IV specifies and verifies the supervisory HA. Section V presents the HA-VDT architecture. Section VI discusses the experimental evaluation. Section VII examines the assumptions and threats to validity. Section VIII concludes the paper.

II. BACKGROUND

A. HYBRID AUTOMATA

Formally, a hybrid automaton [6], [15] is defined as a tuple $H = (L, E, X, \text{Init}, \text{Inv}, f, G, R)$ where:

- L is a finite set of *locations*.
- $E \subseteq L \times L$ is a finite set of *discrete transitions* (or jumps).
- $X = (x_1, \dots, x_n)$ is a finite sequence of real-valued variables. We write $\dot{X} = (\dot{x}_1, \dots, \dot{x}_n)$ to denote the first derivatives during continuous change, i.e., temporal evolution, and $X' = (x'_1, \dots, x'_n)$ to denote the resulting variables after a discrete change. We use v to denote an evaluation of the variables in X (thus, $v \in \mathbb{R}^n$). Notice that, n is the dimension of H .
- Init , Inv and f are labeling functions that assign a predicate to each discrete location $l \in L$. Each initial condition $\text{Init}(l)$ is a predicate whose free variables are from X . Each invariant condition $\text{Inv}(l)$ is a predicate whose free variables are from X . Each flow condition $f(l)$ is a predicate whose free variables are from $X \cup \dot{X}$. E.g., if v is a variable, a flow condition could be $\dot{x} = 5 + 0.1 \cdot x$.
- G is a labeling function that assigns a predicate to each edge $e \in E$, determining if a transition can be fired. Each $G(e)$ condition is a predicate function whose free variables are from X .
- R is a labeling function that assigns a predicate to each edge $e \in E$, determining how variables evolve after the discrete transition. Each reset $R(e)$ is a predicate function whose free variables are from $X \cup X'$.

A *run* of H starts from any l, x such that $\text{Init}(l)[X := x]$ and $\text{Inv}(l)[X := x]$ hold. The control modes represent discrete configurations of the system, while the continuous variables evolve according to the mode-dependent flow conditions as long as the associated invariants hold. When a jump condition

¹<https://carla.org>

becomes satisfied and its discrete event is issued, the system may take the corresponding control switch, thereby changing location and updating the continuous variables. Notice that, under this standard semantics, time is dense and real-valued: within each location, the variables in X evolve continuously according to the flow condition $f(l)$, while discrete jumps are instantaneous. Consequently, any automaton defined according to this formalization inherits a *continuous-time* semantics by construction, rather than a discrete-time approximation of the underlying dynamics.

The HA formalism is widely used because it supports formal reasoning about safety and liveness properties: once the model is defined, its evolution can be analyzed to verify that the system always guarantees some properties. In practice, verification tools approximate the system possible trajectories to check whether unsafe situations can occur. Although some analyses may be computationally demanding, the approach offers valuable guarantees about system behavior under well-defined assumptions [16], [17].

Within the proposed HA-VDT architecture, the HA forms the verified supervisory layer that governs vehicle behavior. By mapping real-time sensor data into the automaton modes and transitions, the system ensures that every control decision, e.g., braking in response to a detected pedestrian, remains consistent with formally established safety constraints.

B. VEHICLE DIGITAL TWIN

A VDT is a dynamic, data-driven virtual representation of a physical vehicle that evolves continuously through real-time synchronization with onboard sensors, control units, and connected systems [7], [8], [9]. Unlike static simulation models, a VDT mirrors the physical system state as it operates, enabling online monitoring, prediction, and optimization under changing environmental conditions [10]. Modern VDTs integrate technologies such as IoT, AI, and cloud-edge computing to maintain a high-fidelity link between the physical and digital domains, supporting proactive functions, fault detection, performance optimization, and adaptive control across the entire vehicle lifecycle [18], [19].

Within this landscape, it is important to distinguish the HA-VDT from two related but architecturally distinct concepts. A *runtime monitor* observes the execution traces of a system and raises alerts when a property is violated, but cannot prevent the violation from occurring and does not govern control decisions [20]. A *model-based controller* uses a model to generate commands but is not necessarily formally verified and does not maintain a continuously synchronized virtual replica of the physical system [21]. Existing VDT architectures for autonomous vehicles largely fall into one of these two categories: they either act as sophisticated monitoring and prediction environments [22], [23], or they apply formal reasoning to scenarios or performance assumptions rather than to the operational controller itself [24].

The HA-VDT differs from both. Within the proposed framework, the DT serves as the computational backbone

that fuses perception and control, while the HA embedded in its modeling core acts as a *verified supervisory controller*: its discrete modes, guards, and invariants have been formally proved to satisfy determinism, deadlock-freedom, and bounded reaction time, and these guarantees are enforced at every control cycle during operation. The VDT thus transitions from a descriptive or monitoring entity into a verified operational one, capable of enforcing safe behavior by construction rather than merely detecting its absence after the fact.

III. RELATED WORK

The safe operation of autonomous vehicles requires solving two complementary problems: proving that control logic satisfies safety properties under all admissible conditions, and synchronizing this verified logic with real-time sensor data in dynamic environments. Existing VDT architectures address the synchronization side but leave the first problem to empirical validation; existing formal verification approaches address safety properties but operate offline and assume idealized inputs. Their integration for runtime supervisory control remains an active research challenge.

A. FORMAL METHODS FOR AUTONOMOUS VEHICLE SAFETY

Research on formal methods for autonomous vehicles has largely centered on reachability analysis, invariant sets, and viability theory, typically applied at design time to characterize safe operating regions or to validate trajectories. Several classical approaches fall into this category. Falcone et al. [25] combine reachability and set-invariance theory with online driver-model identification to predict lane-departure risks, providing anticipatory threat assessment but relying on offline analysis rather than runtime enforcement. Nilsson et al. [26] similarly use viability theory and robust reachability to compute safe, unsafe, and conflict sets under bounded disturbances, generating collision-avoidance guarantees via constrained optimization with quantified robustness. Kianfar et al. [27] extend invariant-set techniques to adaptive and cooperative cruise control, deriving minimal safe distances and maximal asymptotic safe sets for longitudinal interactions. Across these researches, safety is characterized through offline computation: safe regions are defined, but the results are not executed directly as a supervisory controller.

In contrast, our HA-VDT embeds a formally verified HA into the runtime pipeline, meaning that formal guarantees become enforceable decisions that operate continuously under noisy perception, aligning with selected ISO 26262 functional-safety principles.

A second line of research focuses on online reachability-based verification. Althoff and Dolan [28] introduce real-time reachability analysis to over-approximate future occupancy sets for the ego vehicle and surrounding agents, thereby enabling online collision-avoidance verification without relying solely on simulation. Orzechowski et al. [29] combine

Reachable Set Analysis with Responsibility-Sensitive Safety to verify planned trajectories in merging and crossing scenarios, offering strong plan-level guarantees but only before execution. These approaches provide valuable online assessments but stop short of embedding a persistently verified controller: their logic is used to validate/filter trajectories, not to govern the operational system. In contrast, HA-VDT advances this paradigm by implementing a controller whose discrete modes and transitions have been verified through SMT-based encoding of FOL and LTL reasoning; thus, the verified logical structure governs real-time decisions throughout execution rather than being checked episodically or at planning time.

A third group integrates prediction or mode-switching into reachability frameworks. Li et al. [30] cluster human driving behaviors into several modes and precompute Backward Reachable Tubes to reduce conservatism in intersection and roundabout scenarios. Their guarantees remain probabilistic and depend on prediction accuracy, and the mode-switching logic itself is not formally verified. Söntges and Althoff [31] compute over-approximated reachable sets to guide trajectory planning around obstacles and can prove infeasibility when no safe trajectory exists, yet their framework remains offline, lacks standardized validation, and exhibits multi-second runtimes unsuitable for deployment.

In contrast, HA-VDT replaces these offline or prediction-driven approaches with a supervisory controller whose discrete modes and guard conditions are formally verified, allowing reachability-based guarantees to influence runtime decisions rather than only design-time analysis.

B. DTs FOR AUTONOMOUS VEHICLE OPERATION

DTs have become central to evaluating autonomous driving systems, yet most contributions employ DTs as sophisticated testing environments rather than as formally verified runtime supervisors.

A first group of work focuses on high-fidelity simulation platforms designed to maximize realism and scenario coverage. Yu et al. [32] use fault injection, latency analysis, and hardware-in-the-loop integration to expose failure points in an autonomous driving stack, while Liang et al. [33] introduce DTTF-Sim to reconstruct large-scale traffic with statistically calibrated speed and density distributions. Although both systems offer extensive empirical insight, neither incorporates a formal controller model or guarantees that untested scenarios cannot lead to unsafe behavior. Similar limitations appear in integrated simulation environments such as the CARLA-SUMO-CAVE co-simulation [34], which enables realistic V2P accident recreation, but provides no invariants, no verified temporal properties, and no mechanism to enforce safety. Even full-stack DTs such as the system proposed in [35], which includes realistic sensor simulation and traffic replay, treat safety strictly as an empirical measure of reliability across scenarios rather than as a property proved for the controller itself. Field-validated DTs follow the same pattern. The work done in [36] couples

a vehicle and a DT through a 5G link to compare AEB behavior in physical and virtual settings, and Fu et al. [37] replicate real intersections to compute *TTC*-based warnings. Despite their realism, the guarantees they provide remain scenario-bound: controller dynamics are not formalized, temporal correctness is not analyzed, and the DT does not restrict the space of admissible behaviors. Likewise, the B5G-MEC system [38] demonstrates low latency and accurate perception for campus-scale coordination, but once again delivers only measurement-based, not provable, assurances. Campolo et al. [22] propose an edge-based vehicular digital twin (VDT) that leverages containerized microservices, OMA-LwM2M semantic models, and MQTT southbound interfaces for real-time bidirectional synchronization of on-board sensor and CAN-bus data with an edge node. The framework aims to support monitoring, prediction, and safety applications (e.g., cooperative perception and forward collision avoidance) while keeping the communication and computation footprint low. However, safety is addressed only through data-driven monitoring and prediction; the architecture does not embed a formally verified supervisory controller, nor does it provide SMT-based guarantees on determinism, absence of deadlocks, or bounded reaction time.

A second set of studies relies on data-driven or predictive models. Silva et al. [39] examine vulnerable road users safety via surrogate indicators, e.g., *TTC*, Xu et al. [40] integrate SUMO and CarMaker to measure safety under varying environmental conditions, and Gu et al. [41] combine DT synchronization with pedestrian-intent prediction. These researches excel at analyzing risk and complex interactions, but they ultimately define safety in statistical or model-accuracy terms. None formalizes controller logic, proves reachable-state constraints, or derives bounds on reaction time.

A final group supplements DTs with runtime monitoring. Awaad et al. [20] pair a DT with a temporal-logic monitor (TP-DejaVu) to detect violations of selected properties inspired by ISO 26262. While this approach can flag problematic behavior in simulation, it cannot guarantee that violations will never occur, since the autonomous controller (Apollo) is not formally specified and the DT functions only as a source of observable traces rather than as a verified supervisor.

Across these DT frameworks, safety remains empirically assessed rather than formally enforced, since the controller logic is neither specified nor verified. By contrast, HA-VDT incorporates a formally defined supervisory controller into the DT, providing determinism and verifiable safety conditions, without relying solely on scenario-based evaluation.

C. INTEGRATION OF FORMAL METHODS AND DTs

Efforts to integrate formal methods with DTs typically focus on using verification to validate adaptation decisions, deriving formal models from DT behavior for offline checking, or applying formal reasoning to scenarios rather than to the operational controller itself. Wright et al. [42] exemplify the

first category by employing verified monitoring within a DT-based self-adaptive architecture to validate prospective reconfigurations before they occur; while effective for predictive assessment, the verified model acts as an evaluator rather than as the controlling logic. Xavier et al. [43] represent approaches that extract discrete formal models from DT simulation traces and verify them offline through tools such as NuSMV, providing valuable compliance checking but no runtime enforcement and lacking the hybrid expressiveness required for autonomous vehicles. Hou et al. [44] direct formal verification toward the quality of automatically generated test scenarios, strengthening SOTIF-oriented virtual validation but focusing on the test environment rather than on the controller. The SOTERIA framework [24] further broadens this landscape by verifying worst-case computational latencies to support safety assurance, though its guarantees concern performance assumptions rather than the logic governing system behavior. Barbie and Hasselbring [21] formalize the digital twin concept and introduce the notion of Digital Twin Prototypes (DTP) using Object-Z notation together with UML class diagrams. They define rigorous specifications for physical twins, digital models, digital shadows, digital threads, digital templates, digital twins, and DTPs, aiming to support automated integration testing and agile verification/validation of embedded software in cyber-physical systems within CI/CD pipelines. While the work advances formal understanding and design-time testing of DTs, the formalization remains at the conceptual and development level; it does not embed a runtime supervisory controller with verified discrete transitions, invariants, or LTL properties inside an operational Vehicle Digital Twin.

Across these integration efforts, verification typically remains outside the operational control loop. HA-VDT shifts this relationship by placing a verified HA directly in the DT runtime layer: the formally checked logic governs system behavior rather than merely evaluating it.

D. POSITIONING AND CONTRIBUTION

This paper introduces the HA-VDT architecture to address this integration challenge by embedding formal verification directly into the operational safety layer of the DT. Our work is positioned by the following contributions. First, we propose a unified VDT architecture where a formally verified HA functions as the real-time supervisory controller. *Unlike existing VDT efforts, which treat safety empirically through monitoring or scenario-based prediction, the HA-VDT embeds formal guarantees directly into the operational loop, transforming the DT from a descriptive tool into a runtime safety enforcer with by-construction assurances.* This is intended to augment the DT capabilities, moving its role beyond a purely predictive tool to include a mechanism for continuous, formally assured safety enforcement during operation. Second, we demonstrate the use of formal verification techniques, including FOL and LTL modeling and subsequent SMT-based model checking, to verify key functional safety properties. These properties

are foundational requirements for high-assurance systems and are shown to align with expectations from automotive safety standards (e.g., ISO 26262). Finally, the HA, with its explicit modes, guards, and invariants, provides an inherently interpretable model for safety logic. This clarity can support analysis and regulatory compliance by showing the exact logical path taken to arrive at a safety-critical decision. In summary, as it can be seen in Table 1, the HA-VDT framework aims to present a novel, integrated architecture that can effectively combine a VDT with a formally verified, real-time supervisory controller, enhancing both safety and interpretability of autonomous vehicle control systems.

IV. HYBRID AUTOMATON FOR PEDESTRIAN PROTECTION

To enforce safety supervision within the VDT, we define a perception-based *Hybrid Automaton* (HA) that formally captures the discrete logic, continuous evolution, and transitions. The automaton provides a verifiable controller layer within the VDT, ensuring that safety requirements hold even under sensor uncertainty. For each element in the HA definition in Section II, we present the corresponding formalized instance. Notice that, for brevity, we do not provide the formalization for E , G , R and the predicates used in the following section. However, for the interested reader, the complete mathematical specification is available at [14].

Definition 1 (Discrete Locations): The finite set of discrete locations L is defined as follows.

$$L = \{\text{Normal, Throttle, SoftBrk, EmergencyBrk}\},$$

Each discrete mode encodes a specific scenario:

- **Normal:** no detected/imminent pedestrian threat, thus no action required.
- **Throttle:** pedestrian crossing with TTC in the safe-risky range; throttle control is reduced.
- **SoftBrk:** pedestrian crossing with TTC in the risky-critical range; gradual braking is applied
- **EmergencyBrk:** pedestrian crossing under critical TTC conditions; immediate full braking is executed.

Definition 2 (State Variables): The automaton state variable sequence is defined as follows.

$$X = (C_0, \dots, C_{n-1}, TTC_0, \dots, TTC_{n-1}, \\ cs_0, \dots, cs_{n-1}, s_d, s_c, t)$$

where:

- n : the automaton dimension;
- C_i , with $i = 0, \dots, n-1$: confidence of the pedestrian detection in the i -th frame;
- TTC_i with $i = 0, \dots, n-1$: estimated time-to-collision in the i -th frame;
- cs_i with $i = 0, \dots, n-1$: pedestrian crossing indicator in the i -th frame;
- s_d, s_c : timers for detection staleness, crossing staleness, respectively;
- t : time since the last sensor update.

TABLE 1. Grouped comparative analysis of formal verification and DT integration approaches for autonomous vehicle safety.

Refs.	Key Focus	Limitations	HA-VDT Contribution
(A) Reachability and Safe-Set Methods			
[25]–[27], [31]	Predictive or design-time safety analysis using reachability, viability, or invariant sets for collision avoidance and cruise control.	Offline or episodic verification; no runtime supervisory control; cannot enforce guarantees under perception uncertainty.	Real-time safety enforcement through an embedded, formally verified hybrid automaton with ISO 26262 alignment and explicit guarantees under noise.
(B) Online or Real-Time Reachability			
[28]–[30]	Online reachable-set estimation, RSS-based verification, or prediction-based mode selection to validate trajectories.	Plan-level or episodic checks; probabilistic or prediction-dependent guarantees; mode-switching not formally verified.	Continuous supervisory control with SMT-verified properties, certified mode transitions, and worst-case reaction-time guarantees.
(C) DT-Based Testing, Stress Testing, and Co-Simulation			
[22], [32]–[41]	DT platforms for large-scale testing, stress testing, VRU risk mapping, intention prediction, co-simulation, and scenario-based evaluation.	Purely empirical or statistical validation; no worst-case reasoning; no formal controller; relies on scenario coverage rather than guarantees.	Verified supervisory controller embedded in the VDT, enforcing invariants, reachable-set conditions, and bounded reaction-time properties.
(D) Formal Methods Integrated With DTs			
[21], [24], [42]–[44]	Formal verification used for adaptation monitoring, offline model checking of DT-derived models, scenario verification, or timing-assurance analysis.	Formal reasoning applied to adaptation, scenarios, or performance—not the runtime controller; models are predictive or offline only.	Operational safety via a formally verified HA acting as real-time supervisory controller, providing by-construction guarantees and interpretable logic.

The choice of the state variables X is driven by the necessity of *robustness against sensor noise*. The variables are partitioned into buffered sensor measurements (set of C_i , TTC_i and cs_i), which store the most recent n frames to enable a *consensus-based safety logic* that filters out spurious readings. Timers s_d , s_c provide a crucial safety margin by tracking the staleness of the detection and crossing consensus. Specifically, they allow previous detections or crossing states to *remain valid for a bounded period* even if the data is transiently lost (e.g., due to a brief occlusion), preventing potentially unsafe scenarios. This dual approach ensures the discrete safety location transitions are triggered only by *verifiably persistent threats* and not by momentary sensor glitches, thereby bridging the gap between high-frequency perception and reliable control. The variable t serves as a data-driven guard: it forces the automaton to delay any discrete transition until a new sensor frame is available, ensuring all state changes are based on the freshest possible data.

Definition 3 (Initial State): Being $n = |X|$, i.e., the number of variables in X , for any $v \in \mathbb{R}^n$, Init holds if v is:

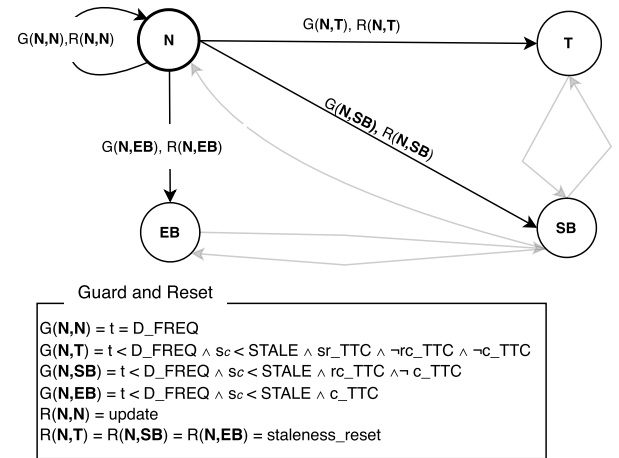
- $C_i = 0$, $cs_i = 0$, $0 \leq i < n$;
- $TTC_i = NO_TTC$, $0 \leq i < n$;
- $s_d = STALE$, $s_c = STALE$;
- $t = 0$.

NO_TTC and $STALE$ are sentinel values to denote a safe TTC and stale data.

Definition 4 (Continuous Evolution): For any $l \in L$, the flow condition $f(l)$ is the same and is defined as:

- $\dot{C}_i = 0$, $\dot{TTC}_i = 0$, $\dot{cs}_i = 0$, $0 \leq i < n$;
- $\dot{s}_d = 1$, $\dot{s}_c = 1$, $\dot{t} = 1$.

In our settings, C_i , TTC_i , cs_i do not change over time as they are updated by discrete events (i.e., those formalized in R), while s_d , s_c , and t grows at a fixed rate.

**FIGURE 1.** Guards and resets from Normal state (N = Normal, T = Throttle, SB = SoftBrk and EB = EmergencyBrk).

Definition 5 (State Invariants): The function $Inv(l)$ is defined as follows.

$$Inv(\text{Normal}) = t < D_FREQ \wedge$$

$$(s_d \geq STALE \vee s_c \geq STALE \vee s_TTC \vee doubt)$$

$$Inv(\text{Throttle}) = t < D_FREQ \wedge s_c < STALE \wedge$$

$$((sr_TTC \wedge \neg rc_TTC \wedge \neg c_TTC) \vee doubt)$$

$$Inv(\text{SoftBrk}) = t < D_FREQ \wedge s_c < STALE \wedge$$

$$((rc_TTC \wedge \neg c_TTC) \vee doubt)$$

$$Inv(\text{EmergencyBrk}) = t < D_FREQ \wedge s_c < STALE$$

In our setting, detections occur every D_FREQ milliseconds; thus, the condition $t < D_FREQ$ forces the automaton to exit the current state and trigger the corresponding transition to handle the arrival of new detections. This ensures state updates based on the most recent sensor

information, allowing timely responses to changing threat levels. Intuitively, the invariants enforce the following properties. In **Normal**, one of the following conditions must hold: (i) no recent pedestrian has been detected (i.e., $s_d \geq \text{STALE}$); (ii) no recent crossing pedestrian has been detected (i.e., $s_c \geq \text{STALE}$); (iii) a pedestrian has been detected at a safe *TTC* (i.e., s_TTC must hold); or (iv) the consensus in the *TTC* buffer has not been reached (i.e., *doubt* predicate must hold). In **Throttle** and **SoftBrk** it must hold that: (i) a crossing pedestrian has been recently detected (i.e., $s_c < \text{STALE}$); (ii) a pedestrian has been detected at a safe-risky or risky-critical *TTC* (i.e., sr_TTC for **Throttle** or rc_TTC for **SoftBrk** must hold); or (iii) the consensus in the *TTC* buffer has not been reached (i.e., *doubt* predicate must hold). Finally, in **EmergencyBrk**, it must hold that a crossing pedestrian has been detected recently.

For brevity, we visually represent, for **Normal** location, edges E , guards G and discrete events R in Figure 1. Guards (marked with G) state that we can reach **Throttle**, **SoftBrk** and **EmergencyBrk**, if no new detection arrives (i.e., $t < D_FREQ$), a crossing pedestrian has been detected (i.e., predicates $s_d < \text{STALE}$ and $s_c < \text{STALE}$) at different threat levels (predicates sr_TTC , rc_TTC and c_TTC). Notice that, the majority of other guards are built upon those represented in Figure 1, e.g., $G(\text{Throttle}, \text{SoftBrk}) = G(\text{Normal}, \text{SoftBrk})$.

The function R , describing discrete evolution of variables, associates the following predicates to edges:

- for any $v, v' \in \mathbb{R}^n$, **update**, occurring in self-cycle edges, holds if v' contains a new detection with C , *TTC* and cs buffers shifted by one position and $t' = 0$. All other elements remain unchanged;
- for any $v, v' \in \mathbb{R}^n$, **staleness_reset**, occurring at any location change, holds if v' has $s'_d = 0$ and $s'_c = 0$, in case a pedestrian is detected and crossing, respectively. All other elements remain unchanged.

A. SAFETY PROPERTIES AND VERIFICATION

To ensure the correctness of our HA, we define a set of *safety and liveness properties* that capture the intended behavior under all possible detection scenarios. These properties are formalized using *First Order Logic* (FOL) and *Linear Temporal Logic* (LTL). As we will see, in our formalization, we use standard LTL operators to express temporal properties over the hybrid automaton. The operator G denotes that a property holds forever in the future behavior, while X indicates that a property holds in the immediate next state. For notational convenience, we write $X^k \varphi$ to indicate that φ holds after k consecutive discrete transitions. This shorthand is equivalent to nesting Xk times, i.e., $X(X(\dots(X(\varphi))\dots))$. Using this notation allows concise specification of properties, e.g., bounded reaction times, without ambiguity in discrete-time evolution.

We now define the predicate **legal**, denoting the admissible values that state variables can assume. Notice that, **legal** always holds throughout the entire automaton evolution.

Definition 6 (Legal Values): Being $n = |X|$, for any $v \in \mathbb{R}^n$, the predicate **legal** holds if v is:

- C_i, TTC_i, cs_i are realistic sensor values, i.e., $C_i \in [0, 1]$, $TTC_i \geq 0$, $cs_i \in \{0, 1\}$, $0 \leq i < n$;
- $D_FREQ \cdot h \leq s_d \leq D_FREQ \cdot (h + 1)$ and $D_FREQ \cdot k \leq s_c \leq D_FREQ \cdot (k + 1)$, where h, k are the smallest indexes in the sequences (i.e., C_i and cs_i) where the predicates to acknowledge detection and crossing statuses hold; if they never hold, then $s_d = s_c = \text{STALE}$. For example, if $cs_0 = 0$, $cs_1 = 0$ and $cs_i = 1$ for $2 \leq i < n$, then $k = 2$;
- $0 \leq t \leq D_FREQ$.

Property 1 (Determinism): Being $n = |X|$, for any legal $v \in \mathbb{R}^n$ and for each $(l, l'), (l, l'') \in E$ with $l' \neq l''$, either the invariant holds or at most one transition can be fired. Formally,

$$\begin{aligned} & \forall (l, l'), (l, l'') \in E. l' \neq l'' \\ & \forall v \in \mathbb{R}^n. \text{legal}[X := v] \\ & \Rightarrow \neg(G(l, l')[X := v] \wedge G(l, l'')[X := v]) \end{aligned}$$

and

$$\begin{aligned} & \forall (l, l') \in E, \forall v \in \mathbb{R}^n. \text{legal}[X := v] \\ & \Rightarrow \neg(G(l, l')[X := v] \wedge \text{Inv}(l)[X := v]) \end{aligned}$$

Explanation: This property enforces deterministic behavior, guaranteeing that sensor readings and control logic always lead to a unique and predictable actuator command. By preventing conflicting transitions, the system ensures safe and reliable interactions between the software controller and the physical vehicle dynamics.

Property 2 (Absence of Blocking States): Being $n = |X|$, for any legal $v \in \mathbb{R}^n$ and for each $(l, l') \in E$, either the invariant holds or there is at least one fireable transition. Formally,

$$\begin{aligned} & \forall v \in \mathbb{R}^n. \text{legal}[X := v] \\ & \Rightarrow \left(\bigwedge_{(l, l') \in E} \neg G(l, l')[X := v] \right) \Rightarrow \text{Inv}(l)[X := v] \end{aligned}$$

Explanation: This property ensures that for every discrete location l , either at least one outgoing transition is enabled or $\text{Inv}(l)$ holds. This guarantees that the controller never reaches a state where no valid action can be taken, preventing deadlocks. Consequently, the vehicle always responds safely to sensor inputs, maintaining continuous interactions between software decisions and physical vehicle behavior.

Property 3 (Upper Bound Reaction Time): For any possible execution of the automaton, if we start from **Normal** and *threat*(C_0, TTC_0, cs_0) (i.e., there is crossing close pedestrian) persists, the automaton enters a braking state (i.e., **SoftBrk** or **EmergencyBrk**) in at most k steps. Formally,

$$\begin{aligned} & G \left(\text{Normal} \wedge \bigwedge_{i=0}^{k-1} X^i(\text{threat}(C_0, TTC_0, cs_0)) \right) \\ & \Rightarrow \bigvee_{j=1}^k X^j(\text{SoftBrk} \vee \text{EmergencyBrk}) \end{aligned}$$

Explanation: From Normal, i.e., absence of threats, whenever a close sudden pedestrian is being detected over time, the automaton enters a braking state, i.e., *SoftBrk*, *EmergencyBrk* mode, in at most k steps. According to the automaton design in [14], $k = 4$ is the maximum number of consecutive detection frames that must agree before the automaton enters *SoftBrk* or *EmergencyBrk*. Being that, new frames arrive each $100ms$, reaction time to threat is $400ms$, almost halving human reaction time, i.e., $700ms$ [45].

Notice that, the property above is expressed with LTL, hence it must hold over all execution paths of the HA.

Together, the three properties align with functional-safety practices in the automotive domain. ISO 26262 [4] requires that safety mechanisms not only detect hazards correctly but also react predictably and within bounded time, because unpredictable or delayed responses can themselves be safety hazards. As noted by safety-critical tool vendors, ISO 26262 explicitly demands worst-case execution-time (WCET) estimates and predictable timing behavior in software components. By expressing timing guarantee in LTL in combination with determinism and guaranteed progress, we provide a formally verifiable specification of our pedestrian protection controller that ensures predictability, bounded reaction, and reliability required for functional-safety architectures.

To formally verify that our HA satisfies the defined properties, we encode the automaton and LTL properties into constraints for the Z3 SMT solver [12]. Continuous sensor variables, timers, and discrete states are represented as Z3 real and integer constants, while transitions and invariants are captured as logical formulas over these variables. To guarantee that a property holds *universally*, and not merely that there exists one configuration satisfying it, we check the *unsatisfiability* of its negation, i.e., no configuration satisfies the negation of the property we want to prove. Violations are automatically detected by Z3 as counterexamples, providing concrete models that lead to violating states.

The use of formal verification aligns with ISO 26262 [4] to achieve high levels of assurance against systematic failures. By mathematically proving the absence of deadlocks, determinism, and bounded reaction times, we address the standard requirements for rigorous validation of safety properties. Furthermore, the explicit modeling of the HA and the associated proofs provide the necessary evidence for certification.

V. HYBRID AUTOMATON-BASED VDT ARCHITECTURE

The HA-VDT architecture establishes a closed-loop system integrating AI-based perception with formally verified control in a unified architecture. As depicted in Figure 2, the architecture consists of three modular components: the *Perception Layer*, the *State Manager* and the *hybrid automaton*. Although the present work focuses on pedestrian protection, arguably one of the most safety-critical urban scenarios, the HA-VDT architecture is a general and reusable

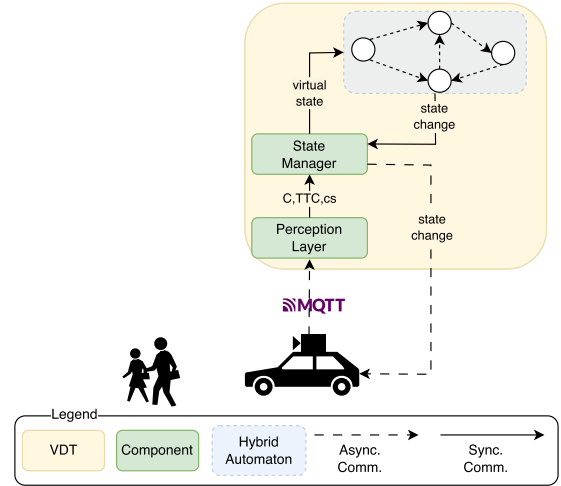


FIGURE 2. HA-VDT architecture overview.

pattern for any safety-critical driving function. Extending the framework to new tasks (e.g., cyclist-aware AEB, adaptive cruise control) requires only the definition of additional discrete modes and guards within the same formally verified HA, together with minimal extensions of the continuous state vector handled by the State Manager. The compositional nature of the verification approach ensures that safety properties already proved for existing modes remain valid.

A. PERCEPTION LAYER

The *Perception Layer* updates the vehicle in real time with information about the surrounding scene, with particular emphasis on the detection and characterization of pedestrians. It relies on two co-located sensors sharing the same mounting point, field of view, and calibration matrix: an RGB camera and a metric depth camera. Because the two sensors are geometrically co-registered by construction, i.e., same intrinsic parameters, same extrinsic pose, no stereo baseline, their output frames are pixel-aligned: pixel (u, v) in the RGB image corresponds to the same scene point as pixel (u, v) in the depth image, with no stereo matching or extrinsic calibration required at runtime. Both sensors are acquired synchronously at the same frequency, guaranteeing that each RGB frame and its associated depth map are temporally consistent.

Pedestrian detection is performed on the RGB stream using the YOLOv8n detector [46], a lightweight single-stage architecture well suited to embedded platforms. For each detected pedestrian, the model outputs a bounding box $B_i = (x_1, y_1, x_2, y_2)$ together with a confidence score $C_i \in [0, 1]$; detections below a confidence threshold are discarded. The bounding box centroid

$$(x_c, y_c) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$$

is used to index the co-registered depth map at the corresponding pixel, yielding a direct metric distance measurement. Because the two sensors are pixel-aligned,

this lookup is geometrically exact: no 3D reconstruction or depth inference from image features is involved. The depth sensor encodes measurements in a 24-bit representation, decoded as

$$d_i^{\text{raw}} = \frac{r + 256g + 256^2b}{256^3 - 1} \times 1000 \text{ m},$$

from which the vehicle front-bumper offset is subtracted to obtain the pedestrian distance d_i from the front of the ego vehicle. Given the current vehicle speed v , the time-to-collision is then $TTC_i = (d_i/v) \times 1000$ ms.

In addition, the system evaluates whether each detected pedestrian lies on a potential collision trajectory via a geometric check against a predefined *Region of Interest* (ROI) aligned with the vehicle heading. The angular half-width of the ROI decreases linearly with distance, mirroring the reduced perceptual and kinematic relevance of far-away subjects:

$$\theta_{\max}(d_i) = \bar{\theta} \left(1 - \frac{d_i}{\bar{d}}\right), \quad 0 \leq d_i \leq \bar{d},$$

where $\bar{d}$ is the maximum distance at which a crossing pedestrian could produce a TTC below the safe threshold TTC_S at the highest considered vehicle speed, and $\bar{\theta}$ is the angular half-width of the vehicle projected at close range, augmented by a lateral safety margin to account for pedestrian body width and lateral positioning uncertainty. The angular deviation $\Delta\psi$ between the pedestrian's image-plane position, converted to an angle using the shared camera calibration matrix, and the vehicle's current heading is computed; a pedestrian is classified as crossing if $|\Delta\psi| \leq \theta_{\max}(d_i)$, producing the binary label $cs_i \in \{0, 1\}$. In the case of multiple simultaneous detections, the Perception Layer forwards to the State Manager only the most critical one: the crossing pedestrian with the smallest TTC , if any, or otherwise the closest non-crossing pedestrian.

The complete end-to-end timing chain from sensing to actuation proceeds as follows. At each perception cycle, the Perception Layer ingests the latest co-registered frame pair, runs inference, computes d_i , TTC_i , and cs_i for each detection, and publishes the resulting payload to the State Manager. The State Manager updates the VDT state buffers and synchronously invokes the HA, which evaluates the active guard and returns a control command to be applied to the vehicle.

B. STATE MANAGER

The *State Manager* maintains the virtual state of the VDT, encoded according to Definition 2. Upon receiving new values: for C , TTC , and cs , it updates the corresponding buffers by shifting existing elements one position to the right and inserting the new measurements at the first position; it updates time-related variables by the elapsed time dt since previously received measurements. The new virtual state is then synchronously fed to the HA, which computes the proper virtual state change to apply, e.g., throttle, brake. Once the state change has been computed, the State Manager reflects

it to the physical object, thereby maintaining consistency with between virtual and physical entities.

C. HYBRID AUTOMATON

The HA component implements the model of Section IV. It operates through a *reactive execution model*. More precisely, once the State Manager sends the current virtual state to the automaton, it reacts as follows:

- 1) it looks for a possibly fireable transition based on the active guard (by Property 1, at any time there is always at most one guard active);
- 2) it computes the next state: if no transition can be fired the current state is still valid (by Property 2, if no transition is fireable, the state invariant still holds). Based on the fired transition, the automaton commands a given state change to the State Manager.

VI. EVALUATION

We now present the results obtained by applying our HA-VDT architecture to the *AEB Pedestrian* scenarios defined by the Euro NCAP 2023 protocol [13] (we just disregard the scenario where the pedestrian is not crossing the vehicle path). The choice of Euro NCAP scenarios allows us to measure the behavior of our system in realistic configurations widely accepted in the industry. The primary goal of this evaluation is not to benchmark against state-of-the-art pedestrian protection algorithms, but to demonstrate that the integrated HA-VDT architecture can successfully execute complex, safety-critical driving functions, while adhering to the formally verified properties, e.g., determinism, bounded reaction time. Independent evaluations of production AEB systems, such as the AsPeCSS project [47], report strong results in non-occluded pedestrian configurations, while also identifying obstructed-pedestrian crossings as the most demanding conditions due to very limited sighting time after occlusion. These findings provide a representative baseline for understanding expected performance trends across the considered Euro NCAP scenarios. The complete code for reproducing the experiments presented throughout this work is available at [14].

A. EXPERIMENTAL SETUP

Experiments are carried out using CARLA 0.9.15 in synchronous mode, employing a sensory pipeline consisting of synchronized RGB and depth cameras producing new frames at 10Hz and sending via Mosquitto MQTT broker.² The ego vehicle is jointly controlled by CARLA autopilot, i.e., steering, and our HA-VDT, i.e., throttling and braking, while its speed is dictated by the scenario under consideration. The HA-VDT runs entirely on an NVIDIA Jetson NX Orin Series edge node, enabling on-board computation under realistic resource constraints. Concretely, our HA-VDT can enact three actions: (i) *throttling acceleration*, i.e., release of the accelerator and, consequently speed reduction via

²<https://mosquitto.org>

engine brake; (ii) *brake*, i.e., active and progressive braking; and (iii) *emergency brake*, i.e., maximum braking stopping vehicle as soon as possible.

Following Euro NCAP directives, the pedestrian and vehicle accelerate until they reach their target speeds and the test starts once the *TTC* between the ego vehicle and pedestrian is 4s.

To test our HA-VDT, we focused on the following four scenarios. The *Car to Pedestrian Farside Adult (CPFA)* scenario, which simulates a pedestrian crossing at a speed of 8km/h from the opposite side of the vehicle direction of travel, fully visible and with a perpendicular trajectory. The *Car to Pedestrian Nearside Adult (CPNA)* scenario represents a nearside crossing, where the pedestrian crosses the road from the vehicle direction of travel side at a speed of 5km/h. The *Car to Pedestrian Nearside Child Obstructed (CPNCO)* scenario, includes a pedestrian initially occluded by a side obstacle, i.e., a parked car, crossing the street from the vehicle direction of travel at a speed of 5km/h. Finally, the *Car to Pedestrian Turning Adult (CPTA)* scenario simulates a pedestrian crossing, from both sides of the road at a speed of 5km/h, during a vehicle turn, combining changes in direction and more complex geometries. For each scenario, we performed 100 independent runs varying ego vehicle speed (we pick 30km/h and 50km/h, as we are in an urban environment) and environmental condition, i.e., day-time and night-time. The only exception is the CPTA test: as required by the Euro NCAP protocol, we run tests only during day-time conditions at speeds of 10km/h and 20km/h. To evaluate the performance of our HA-VDT, we consider the following metrics: (i) the minimum stopping distance $d_{\min}$ of the ego vehicle w.r.t. the pedestrian; (ii) the automaton reaction time, i.e., the time elapsed from the first detection of the crossing pedestrian to the execution of an action; (iii) collision rate; and (iv) Euro NCAP points, which is 1 for avoided collision as well as for collisions with a residual speed lower than 20km/h (for greater residual speeds, we use linear interpolation to compute the score to assign).

TTC thresholds are chosen by combining findings from the traffic-safety literature, human-factors studies, regulatory requirements, and empirical evaluations carried out in Euro-NCAP-based simulations. Early research originating from Hayward [48] and expanded by Vogel [49] shows that *TTC* values in the 1000 – 1500ms range correspond to critical situations in which collision avoidance becomes extremely difficult under realistic braking capabilities. Deveau et al. [50] confirms that a pre-crash *TTC* in naturalistic datasets typically fall in the 1100 – 1400ms interval, reinforcing 1000ms as the practical boundary for last-moment avoidance. Human-factors studies on Forward Collision Warning likewise recommend urgent alerts below a *TTC* of 2000ms and earlier advisory cues around 2500 – 3500ms, consistent with measured driver perception-reaction times, as reported by Puente and Gohl [51]. Regulatory practice in recent NCAP standards also aligns: for example, NHTSA guidelines indicate performance expectations around *TTC* = 2000ms

for warning, and systems are expected to avoid collision under stricter AEB tests [52]. In our architecture, we align with these findings by adopting as thresholds 3500ms, 2000ms, 1000ms, for safe, risky and critical scenarios, respectively.

To ensure robustness against noise, we maintain 10-frame buffers, which capture approximately the last second history at our perception frequency (10Hz) and apply a graded consensus policy checking 20%, 40%, 60% and 80% of the freshest frames for critical, risky-critical, safe-risky and safe, respectively, events. The consensus thresholds were selected a priori to reflect the minimum reliable agreement within our 10-frame buffer. For instance, the 20% threshold for critical events requires confirmation in the 2 most recent frames, providing the fastest possible response that can still overcome a single transient sensor error. Additionally, to account for temporary loss of information due to, e.g., sensor faults, we use an additional consensus factor of 80% applied to all graded consensus thresholds but the critical one.

B. COMMUNICATION TIME ANALYSIS

To quantify the communication latency of the perception-control loop, we conducted an analysis of the propagation delays occurring along the MQTT messaging pipeline and the internal processing time of the HA. For each run, we measured the duration from the moment perception data are published by the vehicle to the broker, through message reception and parsing, to the completion of the automaton decision cycle and the emission of the corresponding control command back to the vehicle. Across a total of 1600 simulations, covering all combinations of scenario configuration, vehicle speed, and environmental conditions, these latency components are aggregated to derive representative statistics of the communication and decision-making overhead. The resulting values, summarizing data reception, hybrid-control processing, and command-return latency, are reported in Table 2.

TABLE 2. Statistical results of MQTT communication times and automaton processing.

Metric in ms	Mean	Min	Max	Std
Send information	21.357	0.189	47.974	12.117
Automaton processing	0.035	0.009	2.006	0.021
Receive information	4.528	0.154	47.985	11.239

Results reported in Table 2 indicate that the dominant contribution to end-to-end latency arises from the communication phases, while the computational overhead of the HA is effectively negligible. Automaton processing time remains consistently less than a millisecond (mean 0.035ms), with minimal variability, confirming that its decision logic introduces no meaningful bottleneck even under the most demanding conditions of the test matrix. In contrast, the send and receive stages show higher dispersion, reflecting the inherent variability of network-based message propagation.

The mean transmission time of $21.357ms$ and reception time of $4.528ms$ remain well within admissible bounds for a closed-loop ADAS pipeline, and their maxima, although an order of magnitude larger, occur infrequently enough not to affect the stability of the control loop. Overall, the aggregated timings demonstrate that the complete perception-decision-actuation cycle operates comfortably within the temporal budget imposed by the $10Hz$ perception stream, ensuring that the HA-VDT can respond with sufficient reactivity to evolving traffic situations. Although MQTT is not the most performant communication protocol for low-latency automotive applications, the measured latencies remain acceptable for our closed-loop setup. Consequently, when deploying the same architecture over automotive Ethernet, whose bandwidth, determinism, and QoS mechanisms substantially exceed those of MQTT, we expect the system to exhibit even lower communication delays and tighter timing dispersion.

C. CPFA: CAR TO PEDESTRIAN FAR SIDE ADULT

The pedestrian starts on the far side of the road and, although detected early, remains outside the vehicle path for a relatively long portion of the trajectory. This explains why the automaton does not intervene initially: the pedestrian is visible but not yet classified as dangerous. Once the pedestrian steps into the path, the crossing state switches immediately, triggering the appropriate control action.

This behavior explains why the measured minimum distances remain reasonable despite the pedestrian being perceived from very far away: the HA-VDT monitors the pedestrian continuously but does not intervene until the pedestrian actually enters the vehicle's path. As it can be seen in Figure 3, the minimum distance d_{min} in different combinations of speed and environmental conditions remains reasonable. At a speed of $30km/h$, both during the day and at night, the vehicle maintains safe distances (approximately 4 meters), and the autonomous system intervenes in most cases only with the mild brake actions and, in a few instances, also switches to brake. As expected, at $50km/h$, due to action space reduction caused by higher speed, the minimum distance decreases yet remaining reasonable (approximately 1.5 meters). As it can be seen, the system performance remains consistent in both environmental conditions with an expected greater variability of d_{min} values at high speed during night: this is caused by lower lighting making it harder to detect pedestrians.

In Figure 4, we show the reaction time, i.e., the interval between the pedestrian being detected below the risky TTC and the execution of a braking action. Average values range between $320 - 400ms$, with a small difference between the two speeds: at $50km/h$, transitions are faster because the TTC decreases rapidly. Notice that, the higher values (around $400ms$) correspond to transitions toward non critical states (e.g., Throttle), which require a greater amount of agreeing frames. Instead, for SoftBrk and EmergencyBrk, the measured reaction time is, on average, $400ms$, empirically confirming what we formally proved in Property 3.

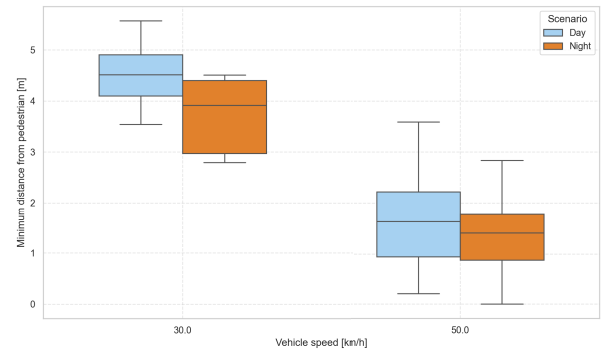


FIGURE 3. Minimum distance d_{min} in the CPFA scenario.

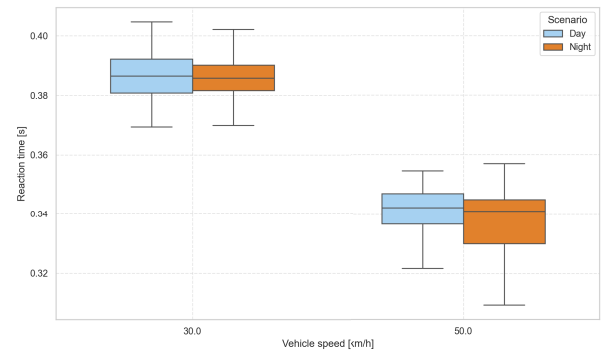


FIGURE 4. Reaction time in the CPFA scenario.

In Table 3 we summarize the main results. As it can be seen, our HA-VDT just hits the pedestrian once over 400 runs (in the worst combination of speed and environmental condition) with a residual speed of $14km/h$. Notice that, according to Euro NCAP score assignment, we get the full point, despite the presence of a collision, thanks to a speed reduction greater than $20km/h$ required by the protocol.

TABLE 3. CPFA scenario results.

Environment	Collisions	v_{res} (km/h)	Score
30km/h (day)	0	–	100
30km/h (night)	0	–	100
50km/h (day)	0	–	100
50km/h (night)	1	14.04 ± 0.00	100

D. CPNA: CAR TO PEDESTRIAN NEARSIDE ADULT

The Car to Pedestrian Nearside Adult (CPNA) configuration poses a substantially harder detection problem than CPFA, as the pedestrian approaches from the nearside, giving the camera a much shorter interval to observe the motion before the pedestrian enters the vehicle path. As a consequence, the system has less contextual information to stabilize the crossing classification, and the moment where the pedestrian genuinely obstructs the trajectory occurs more abruptly. At low speeds this shortened observation window is still

sufficient to avoid braking, but at higher speeds the reduced temporal margin becomes the dominant factor. Night-time conditions exacerbate this effect, as the confirmation of the pedestrian motion is slightly slower, further narrowing the already limited window. These characteristics make late braking more likely and explain the higher number of collisions at 50km/h, even though the residual speeds still meet the Euro NCAP reduction requirement.

The above is reflected in Figure 5, which shows the minimum distance $d_{\min}$ across the different configurations. At 30km/h, no value is reported for $d_{\min}$ because in all runs the system intervenes exclusively with mild brake. At 50km/h, instead, the reduced temporal margin results in smaller and more variable values of $d_{\min}$. The effect is more pronounced at night, where lower illumination slightly delays the confirmation that the pedestrian has entered the vehicle path, producing a wider dispersion of outcomes.

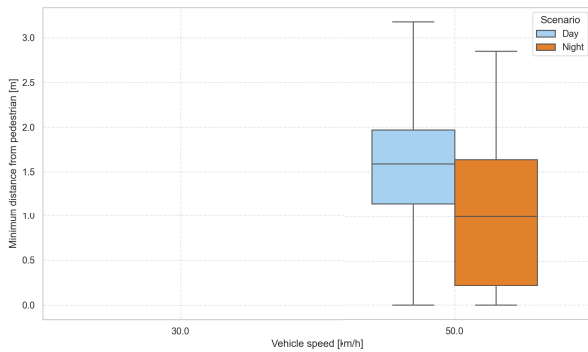


FIGURE 5. Minimum distance $d_{\min}$ in the CPNA scenario.

Figure 6 reports the reaction time under the different conditions. At 30km/h, values are higher (around 500ms), as low level threats require a greater consensus in the automaton buffers; at 50km/h, the TTC drops and braking conditions are reached sooner, resulting in lower reaction times, especially in daylight conditions. As in the CPFA scenario, the highest values correspond to non critical transitions, while for EmergencyBrk and SoftBrk the measured reaction times consistently remain below the 400ms bound.

Table 4 summarises the results. Compared to CPFA, the number of collisions increases at 50km/h, particularly at night, where the combination of reduced lighting and shorter lateral distance reduces the available reaction window. Nevertheless, whenever a collision occurs, the system consistently achieves a significant reduction in speed.

E. CPNCO: CAR TO PEDESTRIAN NEARSIDE CHILD OBSTRUCTED

The Car to Pedestrian Nearside Child Obstructed (CPNCO) scenario represents the most critical condition: a pedestrian, initially occluded by a parked vehicle, suddenly appears. The behavior of the system is shown in Figure 7: at a speed of 30km/h, despite the delayed appearance, the system is able

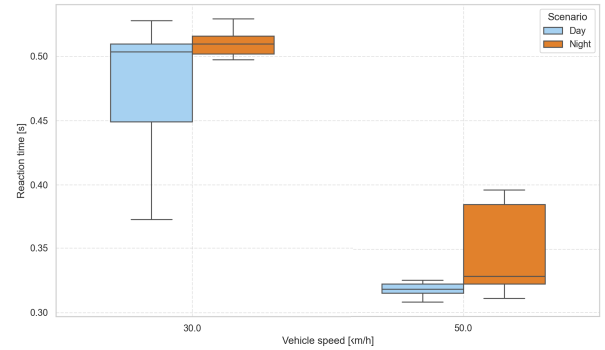


FIGURE 6. Reaction time in the CPNA scenario.

TABLE 4. CPNA scenario results.

Environment	Collisions	v_{res} (km/h)	Score
30km/h (day)	0	–	100
30km/h (night)	0	–	100
50km/h (day)	2	6.45 ± 6.72	100
50km/h (night)	7	14.59 ± 4.56	100

to maintain relatively high final distances of approximately 2.5 – 3m; at 50km/h, in day-time tests, the system still manages to activate effective braking, maintaining an average $d_{\min}$ greater than 1.5m, while in night-time tests, lower lighting further delays the confirmation of the threat by few frames. Such a delay is enough to reduce $d_{\min}$ to values close to zero, resulting in increased variability and collisions.

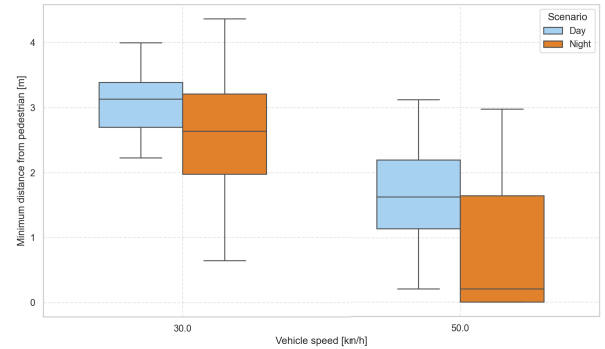


FIGURE 7. Minimum distance $d_{\min}$ in the CPNCO scenario.

In Figure 8, we present the reaction time analysis. At 30km/h, the times are longer than in the previous cases (approximately 400 – 440ms during the day and 500 – 520ms at night), caused by the higher consensus required by low-level threats. At 50km/h, reaction time decreases significantly in day-time runs (approximately 310 – 320ms), as the sudden appearance of the pedestrian at a close distance is a high-level threat scenario that requires low consensus in buffers. At night, HA-VDT reaction is further delayed frames, producing the strong variability observed (320 – 390ms). As in the previous scenarios, the highest values correspond

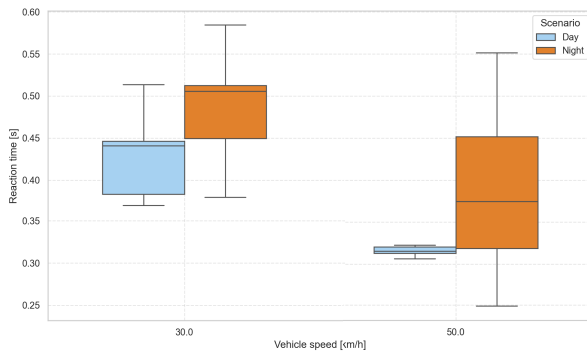


FIGURE 8. Reaction time in the CPNCO scenario.

TABLE 5. CPNCO scenario results.

Environment	Collisions	v_{res} (km/h)	Score
30km/h (day)	0	—	100
30km/h (night)	0	—	100
50km/h (day)	0	—	100
50km/h (night)	36	18.82 ± 2.51	99.65

to non critical actions, while for braking ones the reaction time remains within the 400ms limit proved in Property 3.

Table 5 shows the aggregated analysis. Results, particularly at 50km/h, reflect the same difficulty documented in AsPeCSS [47] under occlusion: the combination of late appearance and reduced visibility results in short *TTC* values, which inherently limits braking effectiveness. Importantly, despite the adversarial conditions, our system achieves substantial *TTC*-aligned speed reduction (average residual 18.82 km/h at night). These results hint at the need for a smarter road infrastructure capable of proactively sharing information with vehicles nearby, providing cooperative perception cues.

Unlike the CPNA scenario, here there are no collisions in day-time executions at 50km/h. The reason is the following. The person exits the parked vehicle and immediately enters an area of strong visual contrast, making their presence more evident to the Perception Layer. In the CPNA case, the pedestrian is visible throughout the approach phase and can be partially confused with the background until they step onto the roadway, delaying confirmation of the crossing status and leaving less margin for intervention.

F. CPTA: CAR TO PEDESTRIAN TURNING ADULT

The Car to Pedestrian Turning Adult (CPTA) scenario is different compared to the previous cases: vehicle does not proceed in a straight line but makes a right turn while the pedestrian crosses the intersection. The following configuration increases the complexity, as the vehicle trajectory changes continuously during the maneuver and the pedestrian position might become critical, depending on the phase of the turn. The Euro NCAP protocol distinguishes between two variants, *farside* and *nearside*. In the former, the pedestrian

comes from the opposite side of the direction of the turn, initially remaining further away from vehicle trajectory (thus, being less critical to manage). In the latter, the pedestrian comes from the same side as the direction of the turn, reducing the time available for the system to intervene and making the task significantly more difficult.

Figure 9 shows the minimum distance d_{min} for all variants and speeds. At 10km/h, the *farside* variant maintains a safety margin greater than 1.5m in all cases, while the *nearside* one, we observe $d_{min} = 0$, due to occasional collisions caused by late detection of the pedestrian in the ROI. At 20km/h, we only have *farside* executions, and the system effectively compensates for the reduction in the time available for intervention and maintains high final distances, with average values of around 2.5m.

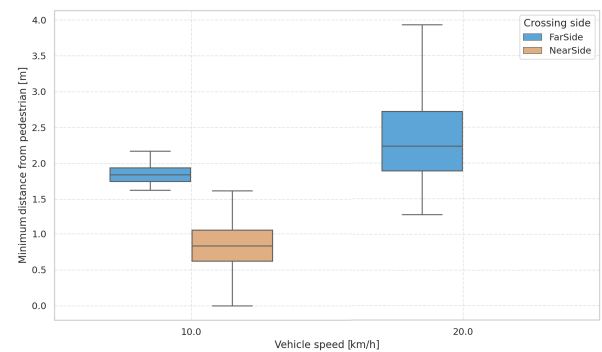


FIGURE 9. Minimum distance d_{min} in the CPTA scenario. The *nearside* variant is only tested at 10 km/h, as the Euro NCAP protocol does not prescribe a nearside configuration at 20 km/h.

Figure 10 shows reaction time. In the *farside* variant, the behavior is stable: at 10km/h, reaction time is close to 500ms, reducing to about 360ms at 20km/h due to the more rapid decrease in *TTC*. In the *nearside* variant, reaction time shows much greater variability, consistent with the strong dependence on the timing the pedestrian becomes visible during the turn. As in previous scenarios, braking actions always comply with the limit established in Property 3.

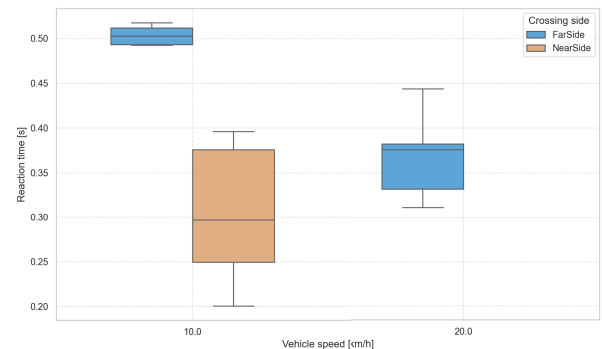


FIGURE 10. Reaction time in the CPTA scenario. The *nearside* variant is only tested at 10 km/h, as the Euro NCAP protocol does not prescribe a nearside configuration at 20 km/h.

The results are shown in Table 6. As dictated by EU NCAP protocol, the score is determined *directly* by the impact speed. The protocol distinguishes between two situations: at 10km/h, in the event of a collision, the score starts at 60 points and decreases according to

$$\text{score} = 60 \cdot \max\left(0, 1 - \frac{v_{\text{res}}}{10}\right).$$

In absence of collisions the protocol assigns the full 100 points, if no impact speed is recorded. At 20km/h for impacts with $v_{\text{res}} \leq 5 \text{ km/h}$, the score decreases by 6 points for each km/h; above this threshold, the reduction continues with a decrease of 10 points for each additional km/h, down to zero. In the CPTA nearside, results exhibit the same performance limitations reported in AsPeCSS [47], where pedestrians enter the sensor field abruptly. In such geometries, Euro NCAP notes that the pedestrian becomes critical comparatively late, leading to shortened *TTC* values and inherently constrained braking margins. The 9 collisions observed at 10km/h, therefore align with the expected behavior of state-of-the-art AEB systems when confronted with late-entry, near-side trajectories. Nevertheless, our low residual speed achieved in these impacts (4.34, km/h on average) confirms that the system is still capable of delivering meaningful mitigation even under these deliberately adverse CPTA conditions.

TABLE 6. CPTA scenario results.

Environment	Collisions	v_{res} (km/h)	Score
Farside 10km/h	0	–	100
Farside 20km/h	0	–	100
Nearside 10km/h	9	4.34 ± 0.41	90.9

VII. THREATS TO VALIDITY

This section discusses the assumptions underlying the formal guarantees of the HA-VDT and the conditions under which those guarantees remain valid at runtime.

A. CONTINUOUS-TIME ABSTRACTION AND SENSING UNCERTAINTY

The HA is specified under standard continuous-time hybrid automaton semantics [6]: timer variables flow at rate 1 in every mode, mode-exit conditions are continuous-time invariants, and discrete jumps are instantaneous. The model therefore captures continuous temporal evolution rather than a discrete-time approximation of vehicle dynamics, and the verification reflects this directly, since time is represented by real-valued variables throughout the Z3 encoding. Periodic frame arrival is not a competing discrete-time nature that must be approximated away, but an ordinary feature expressible within this formalism: between consecutive sensor frames the timers t , s_d , s_c evolve continuously under the flow conditions, while the arrival of a new frame induces an instantaneous discrete jump. This continuous-flow/discrete-jump structure is exactly what hybrid automata are designed to express.

The sampling period D_{FREQ} (100ms in our deployment) enters the model as the bound $t < D_{\text{FREQ}}$ in every mode invariant, forcing the automaton to take a transition at each frame boundary. Consequently, the guarantees established over real-valued time hold at precisely the temporal resolution at which the physical system observes the world, with no discrete-time approximation gap in the temporal model between the verified automaton and the deployed sensing loop.

All residual uncertainties, i.e., sensing noise, occlusion, transient dropouts, timing effects, communication latency, and the inherent limits of handing control back safely, are handled by three complementary mechanisms that operate on top of the verified logic. Each sensor variable is maintained as a sliding window of the n most recent frames, and transitions are triggered only when a minimum fraction of the freshest frames agree on a threat level, filtering transient or spurious detections without weakening the verified properties. The formal properties (Properties 1 and 3) are proved universally over all legal sensor valuations, encompassing the full admissible ranges $C_i \in [0, 1]$, $TTC_i \geq 0$, $cs_i \in \{0, 1\}$. The guarantees thus hold for every value the sensors can report; the relationship between reported values and ground truth (e.g., undetected pedestrians or systematically biased depth readings) lies outside the formal model and is assessed empirically in Section VI. The same boundary applies to human-factor uncertainty: pedestrian behavior enters the model exclusively through the perception-derived quantities C_i , TTC_i , and cs_i , so erratic or unpredictable motion manifests as fluctuating sensor valuations, which are absorbed by the consensus and staleness mechanisms; its empirical impact is covered by the standardized Euro NCAP behavioral patterns.

The timers s_d and s_c track how recently a pedestrian detection and a crossing event were last confirmed. If either signal is transiently lost, due to occlusion, sensor dropout, communication delay, or any other cause, the corresponding timer increments continuously under the automaton flow conditions. Crucially, the staleness mechanism is cause-agnostic: whether a frame is missing because of a physical occlusion, a sensor fault, or a delayed message in the communication pipeline, the effect on the automaton is identical and the response is the same. Once the timer reaches the staleness threshold, the invariant of every active intervention state (Throttle, SoftBrk, EmergencyBrk) is violated, the automaton is forced to transition, and control is unconditionally restored to the driver via Normal. This timeout behavior is explicitly encoded in the automaton invariants and therefore deterministically enforced by the verified supervisory logic, providing a fail-safe that requires no additional runtime mechanism.

B. COMMUNICATION PROTOCOL AND DEPLOYMENT ASSUMPTIONS

The current implementation uses MQTT as the communication substrate between the onboard camera, the Perception Layer, and the State Manager. MQTT is a publish-subscribe

protocol designed for lightweight IoT deployments and does not natively provide the bounded latency, deterministic delivery, or ordering guarantees expected in automotive-grade closed-loop control. This is an acknowledged limitation of the experimental setup. However, two design properties of the HA-VDT limit the practical impact of this choice. First, the supervisory logic is robust to delayed, missing, or out-of-order frames by construction: the staleness mechanism acts as a formal fail-safe, so any communication disruption that prevents a frame from being delivered within the staleness threshold automatically triggers a transition to **Normal**, while the consensus buffer requires agreement across the freshest detections rather than trusting any single message, absorbing reordering without weakening the verified properties. Second, the latency measurements reported in Table 2, aggregated across 1600 simulation runs, show that the end-to-end perception-decision-actuation cycle has a mean latency of approximately 26 ms and a maximum observed latency below 96 ms, both well within the 100 ms inter-frame budget imposed by the 10 Hz perception stream. The verified reaction-time bound (Property 3) accounts for automaton decision latency only; communication overhead is additive but remains negligible relative to this bound in the measured operating conditions. The HA-VDT architecture is also protocol-agnostic by design: the HA and State Manager communicate synchronously, and only the southbound camera-to-broker link uses MQTT. Replacing this link with automotive Ethernet [53] with AVB/TSN scheduling, which provides hardware-enforced latency bounds and guaranteed delivery ordering, would tighten the end-to-end timing envelope and remove the residual uncertainty associated with network-based message propagation, while leaving the verified automaton logic unchanged.

C. SCOPE OF ISO 26262 ALIGNMENT

The paper claims alignment with the functional-safety principles of ISO 26262 [4], not full compliance with the standard. Full compliance would require a complete safety case covering aspects such as freedom from interference between software components, timing and memory partitioning, systematic fault injection and noise injection campaigns, missed-deadline ratio analysis under worst-case bus load, and a documented Automotive Safety Integrity Level (ASIL) decomposition, all of which are beyond the scope of this work.

What the HA-VDT does provide, and what directly maps onto ISO 26262 requirements for software safety mechanisms, are the following. Determinism (Property 1) guarantees that sensor readings always produce a unique, unambiguous actuator command, satisfying the standard's requirement for predictable control-flow behavior. Freedom from deadlock (Property 2) ensures the controller can always respond to sensor inputs, preventing silent failures. Bounded reaction time (Property 3) provides a formally verified worst-case execution bound of 400 ms, providing formal evidence of bounded reaction time and measured evidence that the

automaton processing overhead is negligible in the evaluated setup. Automaton processing overhead, measured at a mean of 0.035 ms across all scenarios (Table 2), confirms negligible computational latency. These verified properties constitute formal evidence toward a safety case and are intended to complement, not substitute, the full qualification process required by ISO 26262 for production deployment.

VIII. CONCLUSION

This paper introduced a Hybrid Automaton-based Vehicle Digital Twin (HA-VDT) that integrates within a VDT a formally verified supervisory controller to enforce safe and interpretable behavior in pedestrian-critical scenarios. The architecture couples a real-time DT, a buffered state representation robust to perception noise, and a hybrid automaton whose guards, invariants, and transitions are specified rigorously and verified through SMT-based reasoning. The verification ensures determinism, guaranteed progress, and bounded reaction time—properties directly aligned with the functional-safety principles prescribed by ISO 26262, particularly the requirements for predictable control-flow behavior, absence of ambiguous actuation decisions, and analyzable worst-case reaction times in safety mechanisms. The evaluation on Euro NCAP 2023 pedestrian-AEB scenarios demonstrates that the HA-VDT maintains low and stable reaction times, consistent braking decisions, and high safety margins. The general performance trends we observe (excellent performance in CPFA/CPNA and degradation in the occluded CPNCO case at high speed during night) are fully consistent with external experimental evidence from AsPeCSS [47], which identifies obstructed-pedestrian scenarios as the primary failure mode for current AEB implementations. The consistency between our HA-VDT results and the independent baseline literature strengthens the external validity of our findings and demonstrates that the observed limitations arise from fundamental perceptual constraints rather than design flaws in the supervisory controller. While the current implementation focuses on pedestrian AEB, both the architecture and the verified HA are modular: new discrete modes, continuous variables, guards, and invariants can be added without invalidating existing safety proofs. Future work will extend the framework to additional safety-critical functions (e.g., cyclist detection, V2X-aware emergency braking, unsignalized intersection management) and to cooperative multi-vehicle DTs, further exploiting the compositional nature of HA and runtime monitoring techniques toward ISO 26262.

ACKNOWLEDGMENT

The work reflects only the authors' views; European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

- [1] T. Zhao, E. Yurtsever, J. A. Paulson, and G. Rizzoni, "Formal certification methods for automated vehicle safety assessment," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 1, pp. 232–249, Jan. 2023.

- [2] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE Int. J. Transp. Saf.*, vol. 4, no. 1, pp. 15–24, Apr. 2016.
- [3] J. V. Deshmukh and S. Sankaranarayanan, "Formal techniques for verification and testing of cyber-physical systems," in *Design Automation of Cyber-Physical Systems*. Cham, Switzerland: Springer, 2019, pp. 69–105.
- [4] *ISO 26262: Road Vehicles—Functional Safety*, International Organization for Standardization, Geneva, Switzerland, 2018.
- [5] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Toward verified artificial intelligence," *Commun. ACM*, vol. 65, no. 7, pp. 46–55, Jul. 2022.
- [6] T. A. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. IEEE Symp. Log. Comput. Sci.*, Jul. 1996, pp. 278–292.
- [7] M. Grieves et al., "Digital twin: Manufacturing excellence through virtual factory replication," *White Paper*, vol. 1, no. 2014, pp. 1–7, 2014.
- [8] W. Kritzing, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.
- [9] Y. Lu, C. Liu, K. I.-K. Wang, H. Huang, and X. Xu, "Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues," *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101837.
- [10] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21980–22012, 2020.
- [11] L. Huang, L. R. Varshney, and K. E. Willcox, "Formal verification of digital twins with TLA and information leakage control," 2024, *arXiv:2411.18798*.
- [12] L. De Moura and N. Björner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.* Cham, Switzerland: Springer, 2008, pp. 337–340.
- [13] Euro NCAP. (2023). *AEB/LSS VRU Test Protocol, Version 4.5.1*. Accessed: Nov. 18, 2025. [Online]. Available: <https://www.euroncap.com/media/80156/euro-ncap-aeb-lss-vru-test-protocol-v451.pdf>
- [14] L. Bacchiani and A. Bedei. (2025). *Formally Verified Pedestrian Protection*. Accessed: Oct. 24, 2025. [Online]. Available: <https://github.com/LBacchiani/formal-pedestrian-protection>
- [15] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. S. Sastry, "Dynamical properties of hybrid automata," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 2–17, Jan. 2003, doi: [10.1109/TAC.2002.806650](https://doi.org/10.1109/TAC.2002.806650).
- [16] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Proc. Int. Hybrid Syst. Workshop*. Cham, Switzerland: Springer, 1991, pp. 209–229.
- [17] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proc. Int. Conf. Comput. Aided Verification*. Cham, Switzerland: Springer, 2011, pp. 379–395.
- [18] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.
- [19] Ö. Kaya, L. Bacchiani, A. Melis, R. Presta, C.-T. Lam, G. Pau, and R. Girau, "Vehicle, driver, and road digital twins for connected mobility: A critical review and unified conceptual framework," *Future Internet*, vol. 18, no. 6, p. 277, May 2026.
- [20] T. A. Awaad, H. A. Elged, M. A. Abu-Bakr, S. Y. Fathy, S. H. Ahmed, M. W. El-Kharashi, and M. AbdElSalam, "Safety analysis of automated driving platforms using digital twin simulation and runtime monitoring," in *Proc. DVCon*, 2025, pp. 1–8.
- [21] A. Barbie and W. Hasselbring, "From digital twins to digital twin prototypes: Concepts, formalization, and applications," *IEEE Access*, vol. 12, pp. 75337–75365, 2024, doi: [10.1109/ACCESS.2024.3406510](https://doi.org/10.1109/ACCESS.2024.3406510).
- [22] C. Campolo, G. Genovese, A. Molinaro, B. Pizzimenti, G. Ruggeri, and D. M. Zappalà, "An edge-based digital twin framework for connected and autonomous vehicles: Design and evaluation," *IEEE Access*, vol. 12, pp. 46290–46303, 2024, doi: [10.1109/ACCESS.2024.3382001](https://doi.org/10.1109/ACCESS.2024.3382001).
- [23] S. Almeaibed, S. Al-Rubaye, A. Tsourdos, and N. P. Avdelidis, "Digital twin analysis to promote safety and security in autonomous vehicles," *IEEE Commun. Standards Mag.*, vol. 5, no. 1, pp. 40–46, Mar. 2021.
- [24] K. Wilson, A. A. Arafat, J. Baugh, R. Yu, X. Liu, and Z. Guo, "Soteria: A formal digital-twin-enabled framework for safety-assurance of latency-aware cyber-physical systems," in *Proc. 28th ACM Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, May 2025, pp. 1–11, doi: [10.1145/3716863.3718028](https://doi.org/10.1145/3716863.3718028).
- [25] P. Falcone, M. Ali, and J. Sjöberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1352–1361, Dec. 2011, doi: [10.1109/TITS.2011.2158210](https://doi.org/10.1109/TITS.2011.2158210).
- [26] J. Nilsson, J. Fredriksson, and A. C. E. Ödöblom, "Verification of collision avoidance systems using reachability analysis," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 10676–10681, 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1474667016433100>
- [27] R. Kianfar, P. Falcone, and J. Fredriksson, "Reachability analysis of cooperative adaptive cruise controller," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 1537–1542.
- [28] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.
- [29] P. F. Orzechowski, K. Li, and M. Lauer, "Towards responsibility-sensitive safety of automated vehicles with reachable set analysis," in *Proc. IEEE Int. Conf. Connected Vehicles Expo (ICCVE)*, Nov. 2019, pp. 1–6.
- [30] A. Li, L. Sun, W. Zhan, M. Tomizuka, and M. Chen, "Prediction-based reachability for collision avoidance in autonomous driving," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 7908–7914.
- [31] S. Söntges and M. Althoff, "Determining the nonexistence of evasive trajectories for collision avoidance systems," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, Sep. 2015, pp. 956–961.
- [32] B. Yu, C. Yuan, Z. Wan, J. Tang, F. Kurdahi, and S. Liu, "ADDT—A digital twin framework for proactive safety validation in autonomous driving systems," 2025, *arXiv:2504.09461*.
- [33] Z. Liang, J. Wang, T. Zhang, and X. Yong, "DTTF-sim: A digital twin-based simulation system for continuous autonomous driving testing," *Sensors*, vol. 25, no. 11, p. 3447, May 2025. [Online]. Available: <https://www.mdpi.com/1424-8220/25/11/3447>
- [34] Z. Wang, O. Zheng, L. Li, M. Abdel-Aty, C. Cruz-Neira, and Z. Islam, "Towards next generation of pedestrian and connected vehicle in-the-loop research: A digital twin co-simulation framework," 2022, *arXiv:2212.05090*.
- [35] B. Yu, C. Chen, J. Tang, S. Liu, and J.-L. Gaudiot, "Autonomous vehicles digital twin: A practical paradigm for autonomous driving system development," *Computer*, vol. 55, no. 9, pp. 26–34, Sep. 2022.
- [36] M. Zhang, J. Xu, Y. Gao, D. Shen, and Z. Xu, "Can combined virtual-real testing speed up autonomous vehicle testing? Findings from AEB field experiments," *Commun. Transp. Res.*, vol. 5, Dec. 2025, Art. no. 100216. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2772424725000563>
- [37] Y. Fu, M. K. Turkcan, V. Anantha, Z. Kostic, G. Zussman, and X. Di, "Digital twin for pedestrian safety warning at a single urban traffic intersection," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2024, pp. 2640–2645.
- [38] K. Nonomura, K. Wang, G. Cho, H. Matsuo, J. Nakazato, Z. Li, T. Yu, and K. Sakaguchi, "Proof-of-concept of digital twin for road safety assisted by B5G heterogeneous MEC network," in *Proc. IEEE 21st Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2024, pp. 1110–1111.
- [39] R. M. Silva, G. F. Azevedo, M. V. V. Berto, J. R. Rocha, E. C. Fidelis, M. V. Nogueira, P. H. Lisboa, and T. A. Almeida, "Vulnerable road user detection and safety enhancement: A comprehensive survey," *Expert Syst. Appl.*, vol. 292, Nov. 2025, Art. no. 128529, doi: [10.1016/j.eswa.2025.128529](https://doi.org/10.1016/j.eswa.2025.128529).
- [40] G. Xu, J. Chen, Z. Wang, A. Zhou, M. Schrader, J. Bittle, and Y. Shao, "Enhancing traffic safety analysis with digital twin technology: Integrating vehicle dynamics and environmental factors into microscopic traffic simulation," 2025, *arXiv:2502.09561*.
- [41] X. Gu, W. Duan, G. Zhang, J. Hou, L. Peng, M. Wen, F. Gao, M. Chen, and P.-H. Ho, "Digital twin technology for intelligent vehicles and transportation systems: A survey on applications, challenges and future directions," *IEEE Commun. Surveys Tuts.*, vol. 28, pp. 3235–3271, 2026.
- [42] T. Wright, C. Gomes, and J. Woodcock, "Formally verified self-adaptation of an incubator digital twin," in *Proc. 11th Int. Symp. Leveraging Appl. Formal Methods, Verification Validation Pract. (ISoLA)*, (Lecture Notes in Computer Science), vol. 13704, Rhodes, Greece, T. Margaria and B. Steffen, Eds., Cham, Switzerland: Springer, 2022, pp. 89–109, doi: [10.1007/978-3-031-19762-8_7](https://doi.org/10.1007/978-3-031-19762-8_7).
- [43] M. Xavier, J. Håkansson, S. Patil, and V. Vyatkin, "Plant model generator from digital twin for purpose of formal verification," in *Proc. 26th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2021, pp. 1–4.

- [44] Z. Hou, S. Wang, H. Liu, Y. Yang, and Y. Zhang, "Twin scenarios establishment for autonomous vehicle digital twin empowered SOTIF assessment," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 1965–1976, Jan. 2024.
- [45] M. Green, "How long does it take to stop? Methodological analysis of driver perception-brake times," *Transp. Hum. Factors*, vol. 2, no. 3, pp. 195–216, 2000.
- [46] G. Jocher, A. Chaurasia, and J. Qiu. (2023). *YOLOv8: Ultralytics Object Detection Model*. accessed: 2025-10-24. Accessed: Oct. 24, 2025. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [47] P. Seiniger, A. Hellmann, O. Bartels, M. Wisch, and J. Gail, "Test procedures and results for pedestrian AEB systems," in *Proc. 24th Conf. Enhancement Saf. Vehicles (ESV)*, Gothenburg, Sweden, 2015, pp. 1–22.
- [48] J. Hayward, *Near Misses as a Measure of Safety at Urban Intersections*. Pennsylvania, PA, USA: Transportation and Traffic Safety Center, 1971.
- [49] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Anal. Prevention*, vol. 35, no. 3, pp. 427–433, May 2003.
- [50] D. Deveaux, T. Higuchi, S. Uçar, C.-H. Wang, J. Härri, and O. Altintas, "Extraction of risk knowledge from time to collision variation in roundabouts," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 3665–3672.
- [51] P. Puente Guillen and I. Gohl, "Forward collision warning based on a driver model to increase drivers' acceptance," *Traffic Injury Prevention*, vol. 20, no. sup1, pp. S21–S26, Jun. 2019.
- [52] N. H. T. S. Administration. (2024). *New Car Assessment Program Final Decision Notice—Advanced Driver Assistance Systems Roadmap*. [Online]. Available: <https://www.regulations.gov/document/NHTSA-2024-0077-0001>
- [53] K. Matheus and T. Königseder, *Automotive Ethernet*. Cambridge, U.K.: Cambridge Univ. Press, 2021.



LORENZO BACCHIANI received the Ph.D. degree in computer science and engineering from the University of Bologna, in 2025. He is currently a Research Fellow with the Department of Computer Science and Engineering, University of Bologna. His research interests include behavioral-based methodologies for component interaction, adaptation, and deployment, and digital twin technologies in the automotive domain.

In particular, his work explores the development and integration of vehicle, driver, and road digital twins to improve system reliability and safety.



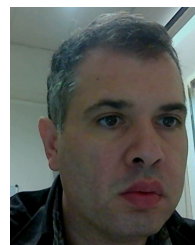
ANDREA BEDEI (Graduate Student Member, IEEE) received the M.S. degree in computer science and engineering from the University of Bologna, Cesena Campus, in 2025, where he is currently pursuing the Ph.D. degree in automotive engineering for intelligent mobility. His research interests include computer vision, machine learning, simulation frameworks for autonomous vehicle safety, and the use of virtual environments for marine robotics and underwater drone experimentation.



ÖZLEM KAYA received the M.Sc. degree in mechanical engineering from Uludağ University, in 2018. She is currently pursuing the Ph.D. degree in automotive engineering for intelligent mobility with the University of Bologna. She is also working on autonomous driving algorithms and digital twin technology.



ROBERTA PRESTA received the M.Sc. degree in telecommunications engineering and the Ph.D. degree in information and automation engineering from the University of Naples Federico II, in 2009 and 2013, respectively, and the Ph.D. degree in humanities and technologies from Suor Orsola Benincasa University of Naples, Italy, in 2021. Since 2013, she has been a Researcher with the Scienza Nuova Interdepartmental Research and Project Design Center, Suor Orsola Benincasa University of Naples, where she is currently an Assistant Professor with the Department of Education, Psychology and Communication. Her research interests include the field of human-computer interaction and deals with the design and assessment of interactive systems in different application domains.



MARIO BRAVETTI is currently a Full Professor with the Department of Computer Science and Engineering, University of Bologna. His research interests include formal methods for distributed systems, concurrency and probability theory, and applications to service-oriented and cloud computing. He has contributed to numerous international research projects and served on editorial boards of international computer science journals. He has authored more than 100 peer-reviewed publications. He has held key roles in major international conferences, including CONCUR, SEFM, SAC, and QEST, and in several international Ph.D. schools and scientific committees.



ROBERTO GIRAU (Senior Member, IEEE) received the M.S. degree in telecommunication engineering and the Ph.D. degree in electronic engineering and computer science from the University of Cagliari, Italy, in 2012 and 2017, respectively. From 2012 to 2020, he was a Researcher with the Department of Electrical and Electronic Engineering, University of Cagliari, developing an experimental platform for the social Internet of Things. He has been an Associate Professor with the Department of Computer Science and Engineering, University of Bologna, since 2021. His main research interests include the IoT and vehicular and human digital twins, with particular emphasis on safety applications.

...