

INSIGHT: Outcome-driven Event Recording for Autonomous Vehicles

Yuxin Wang and Weisong Shi

Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA

{yuxw, weisong}@udel.edu

Abstract—Autonomous vehicles can handle many routine driving scenes, but they still fail or struggle in sparse long-tail situations that are underrepresented in collected driving data. These rare but high-value moments are essential for failure analysis, model improvement, and safety validation. However, most OEMs’ event data recorders only save crash-related reconstruction data. In contrast, continuous recording, like dash cams, captures more evidence but wastes limited edge storage on ordinary driving. Scene-level mining of rare objects, anomalies, and out-of-distribution inputs can help identify potentially difficult scenes, but it does not determine whether the agent actually struggled or produced behavior worth learning from. We present INSIGHT, an online event data recorder that organizes retention around realized vehicle outcomes. INSIGHT detects crash, near-crash, and unstable-control outcomes from onboard signals, then converts trigger activations into fixed crash evidence windows and adaptive, merged, deduplicated non-crash clips. On 220 Bench2Drive routes with TF++ autonomous driving agent, INSIGHT covers 100.0% of crashes and captures 69.5% of non-crash safety evidence where the agent struggles and saves about 81.6% compared with continuous recording. On Raspberry Pi 5, INSIGHT runs in 2.38 ms within a 100 ms cycle, with 0.39%/24.1% mean/peak CPU and 405.3MB/689.6MB mean/peak memory use. The transfer study also shows similar event capture capability, storage efficiency, and lightweight system cost without any INSIGHT configuration tuning for a weaker agent.

Index Terms—event data recorder, autonomous vehicles, edge computing, storage efficiency

I. INTRODUCTION

Autonomous driving has progressed from demonstrations to large-scale road testing and commercial deployment. However, reliable operation in the open human world remains difficult. Real roads are not deterministic testbeds: traffic participants behave unpredictably, road geometry varies across regions, construction and weather change the scene, and rare interactions may only appear after extensive deployment. As a result, even capable autonomous driving systems can still fail or struggle in long-tail situations that are underrepresented in standard training and evaluation data.

A common industry response is to collect more real-world driving data from tested or deployed vehicles. Large autonomy programs use fleet and road-test data to improve driving models, evaluate safety behavior, and investigate failures [1], [2]. Regulators also increasingly require event data recorder (EDR) installation on vehicles to preserve information related to traffic collisions [3]. However, collecting useful driving data is not only a data-scale problem. It is also a systems problem for onboard recording. A vehicle continuously produces high-

bandwidth camera, motion, control, and state streams, while most of these data correspond to routine driving, whereas the most valuable learning and investigative signals are sparse, temporally localized, and difficult to identify online under limited onboard resources.

Existing EDR has different strategies. Currently common in-vehicle EDR provides an important mechanism for crash reconstruction and compliance, but its scope is centered on crash evidence. This is too narrow for autonomous-driving development, where many informative events occur before a collision or without any collision at all. Continuous recording, like dash cams, provides broader coverage, but it spends storage and data movement on long periods of ordinary behavior [4]. Another line of work mines rare objects, anomalies, out-of-distribution (OOD) inputs, or semantically unusual scenes [5]–[7]. These methods are valuable because they identify scenes that may be difficult for an autonomous driving system. However, scene-level mining only indicates potential difficulty; it does not determine whether the deployed agent actually struggled or produced behavior worth learning from. A rare object on the roadside, an unusual vehicle appearance, or an uncommon lighting condition may be handled smoothly by a robust driving policy. In contrast, an ordinary junction, a parked vehicle, a merge, or a pedestrian interaction may still lead to a collision event. This creates a mismatch between scene-level rarity and recording value.

This paper presents **INSIGHT**, an online event data recorder that organizes autonomous-vehicle data retention around realized vehicle outcomes. In this paper, we use *edge* to refer specifically to the event data recorder device, a lightweight onboard recorder that is connected to the vehicle data stream but separate from the main autonomous-driving computer. As shown in Figure 1, this edge recorder receives online sensor, state, and control signals, runs the trigger and recording logic locally, and stores selected evidence clips under its own limited compute, memory, bandwidth, and storage budget.

We use *outcome* to mean the observable behavior of the ego vehicle after the driving system acts in a scene. In this work, the outcomes of interest are crash, near crash, unstable control, and normal driving. A crash is a realized failure. A near crash is an imminent geometric conflict that narrowly avoids collision. Unstable control is a non-crash outcome in which the vehicle remains safe, but its control stream exhibits repeated corrections, hesitations, or loss of smoothness. Nor-

mal driving refers to periods without these failure, risk, or instability signals. INSIGHT uses these realized outcomes as the retention signal to retain compact evidence clips when vehicle behavior indicates that the agent failed or struggled.

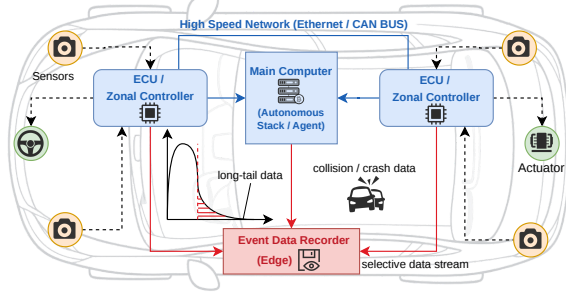


Fig. 1. Centralized vehicle architecture with an edge event data recorder. CAN denotes Controller Area Network, ECU denotes Electronic Control Unit.

INSIGHT has two main components. First, a lightweight on-line trigger module monitors onboard signals and detects crash, near-crash, and unstable-control outcomes during driving. Second, a storage-aware recorder converts trigger activations into canonical event clips. Crash outcomes receive fixed evidence windows to preserve the required context under the regulation. Non-crash outcomes use adaptive windows that follow trigger duration, merge nearby activations from the same episode, and remove redundant clips through persistent deduplication.

We evaluate INSIGHT in closed-loop autonomous-driving workloads using Bench2Drive and two different driving agents. On 220 Bench2Drive routes with TF++, INSIGHT covers 100.0% of crash events and captures 69.5% of non-crash safety events. It reduces 419 online detections to 156 final saved clips, saving about 81.6% of storage while still preserving most safety-relevant non-crash evidence compared with continuous recording. It expands retention beyond collision reconstruction to near-crash and unstable-control behavior with only a modest storage increase compared with crash-only EDR. Deployed on Raspberry Pi 5, INSIGHT runs in 2.38 ms within a 100 ms cycle budget, with low CPU usage and bounded memory cost. The transfer study with the weaker TCP agent shows similar event capture, storage efficiency, and edge deployment behavior without retuning the INSIGHT configuration.

This paper makes the following contributions:

- We formulate autonomous-vehicle recording as an outcome-driven retention problem, addressing the mismatch between scene-level rarity and evidence that the agent actually failed or struggled.
- We design **INSIGHT**, an online recorder that detects crash, near-crash, and unstable-control outcomes from onboard signals and converts them into compact storage-aware evidence clips.
- We evaluate INSIGHT on Bench2Drive and Raspberry Pi 5 with two different agents, demonstrating reliable crash preservation, meaningful non-crash capture, compact storage, low online overhead, and transferability.

II. BACKGROUND AND RELATED WORK

A. Crash-Centered Event Data Recorders

Vehicle event data recorders were designed primarily for crash evidence. They preserve a short interval of vehicle dynamics and safety-system state so that investigators can reconstruct what happened before and during a collision. This objective is reflected in U.S. Part 563, which specifies EDR data elements, survivability, retrieval, and owner-manual disclosure [8]. Recent NHTSA rulemaking extends the timed pre-crash recording period and increases the sampling frequency, improving crash reconstruction fidelity while preserving the same crash-centered scope [3].

This scope is insufficient for autonomous driving development. Prior traffic-safety and criticality studies show that useful safety evidence includes near misses, conflicts, and critical interactions that occur before any realized collision [9]–[11]. A crash-only recorder misses these non-crash learning signals by design. Continuous recording captures them, but creates high storage and bandwidth pressure by retaining long intervals of routine driving, not to mention that a typical autonomous vehicle even generates more than Terabytes of sensor data per day [12]. This motivates selective recording that preserves crash evidence while also capturing other learnable critical events for the autonomous vehicle agent’s further improvement, not just accident analysis.

B. Selective Automotive Recording

There are two representative works on selective automotive recording: Smart Black Box proposes value-driven retention for high-bandwidth automotive data [13], and Smart Black Box 2.0 extends the idea with video-anomaly and action-detection methods for event-of-interest discovery [14]. The original Smart Black Box mainly assumes predefined events are already available and focuses on how to retain them under limited storage. Smart Black Box 2.0 adds learned video-based event detection, but remains centered on some specific anomaly-informed video retention not broad enough to different scenes. And there is no evaluation demonstrating the potential edge deployability with real-time detection and data storage on limited computing resources.

Scene	Realized vehicle outcome			
	Normal	Unstable	Near crash	Crash
Rare event	•••	••	•	•
Uncertainty	•	••	••	••
Out of distribution	••	••	•	•
Visual anomaly	••	••	•	•
Safety critical	•	••	•••	•••

Fig. 2. Relationship between scene-level descriptions and realized vehicle outcomes. The black dot represents the possibility.

C. Scene-Level Corner Cases, Anomalies, and Out-of-Distribution Inputs

Autonomous-driving research uses several related terms for difficult data, including corner cases, edge cases, anomalies,

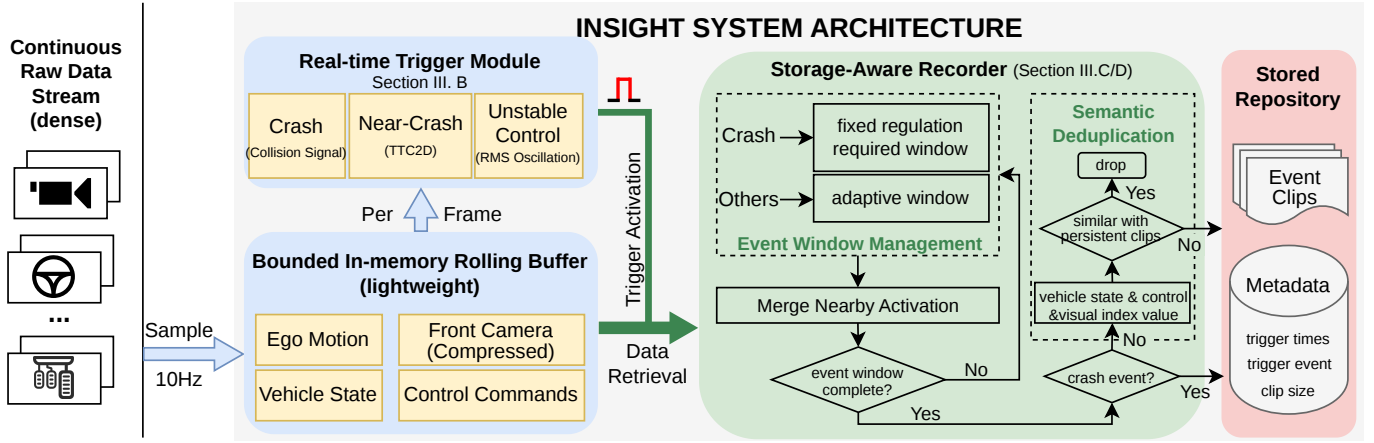


Fig. 3. INSIGHT system architecture. A dense online driving stream is sampled into a lightweight rolling buffer containing compressed camera frames, ego motion, vehicle state, and control commands. The real-time trigger module detects crash, near-crash, and unstable-control outcomes, and the storage-aware recorder converts trigger activations into fixed crash evidence windows or adaptive non-crash windows. Nearby activations are merged, completed windows are deduplicated, and the final retained event clips and metadata are written to the stored repository.

out-of-distribution inputs, long-tail scenes, and safety-critical events [5], [6]. These works study how to discover or define unusual situations that may challenge a driving system. CODA collects real-world road corner cases with image-domain tags, LiDAR alignment, and object annotations [15]. Out-of-distribution detection methods identify samples whose appearance or learned feature distribution differs from training data [16]–[18]. Road-anomaly segmentation methods localize unexpected obstacles or open-world objects in the scene [19], [20].

These methods are valuable for identifying potentially difficult inputs, but scene-level rarity does not mean the agent’s ability to handle the situation [21]. A rare object, unusual vehicle appearance, or uncommon lighting condition may improve dataset coverage or OOD evaluation, but it does not necessarily mean the agent failed or struggled. As shown in Figure 2, different scene types can still produce the same set of realized vehicle outcomes: crash, near crash, unstable control, or normal driving. Conversely, an ordinary junction, merge, parked-vehicle interaction, or pedestrian crossing may still produce near-collision risk or unstable behavior, which are better captured by criticality metrics than by image rarity alone. Thus, scene-level analysis is useful for explaining why an event occurred, but it is not sufficient as the primary online retention rule to reflect agent performance.

D. Trajectory Criticality and Near-Crash Metrics

Traffic-safety research has long used surrogate safety metrics to identify near misses and conflict severity. Time to collision (TTC), post-encroachment time, anticipated collision time, and related trajectory-based measures estimate whether road users are likely to conflict before an actual crash occurs [22]–[25]. Knowledge-driven criticality methods and ontologies further organize hazardous situations using structured traffic concepts and scenario relations [26], [27]. These studies are closely related to our near-crash trigger

because they provide principled ways to reason about pre-crash risk. However, many of them are primarily formulated as analytical metrics that need accurate ground truth or privileged scene parameters, making them hard to implement. INSIGHT adopted this idea but achieved real deployment with only vehicle-generated raw data.

III. SYSTEM DESIGN

INSIGHT is an online event-retention system for autonomous vehicles. It is designed to run continuously beside a driving stack, observe the raw vehicle data, and decide whether the recent data should be retained. The design has two principles. First, the trigger should be based on realized agent outcomes rather than scene rarity alone. Second, the recorder should be storage-aware: it should preserve enough context to make an event useful while avoiding too many storage cost. The overall system architecture of INSIGHT is shown in Figure 3.

A. Runtime Model

INSIGHT runs as an edge recorder inside the autonomous vehicle. It is logically separated from the driving agent, but it receives the same onboard signals needed for event retention, including vehicle state, control commands, camera observations, crash indicators, and surrounding object tracks. This separation allows the autonomy stack to continue operating as the primary control system, while INSIGHT independently monitors realized vehicle outcomes and decides which evidence should be retained.

At runtime, INSIGHT follows a continuous streaming model. Each incoming vehicle sample is appended to a bounded rolling buffer that keeps recent sensor, state, control, and trigger information. The buffer provides pre event context without requiring full route recording. For every sample, the trigger module updates the crash, near crash, and unstable control states. If no trigger is active, INSIGHT only maintains the

rolling buffer and lightweight statistics. If a trigger becomes active, the recorder opens or extends an event window and marks the corresponding time interval for possible retention.

The recorder uses different retention behavior for crash and noncrash outcomes. A crash trigger immediately creates a fixed evidence window with sufficient pre event and post event context. Near crash and unstable control triggers create adaptive windows whose duration follows the observed trigger activity. Nearby activations are merged into one window so that a single driving episode is not fragmented into multiple clips. A window is finalized only after the trigger has remained inactive for a short quiet period, which prevents premature closing during intermittent risk or oscillatory control.

Once an event window is finalized, INSIGHT materializes the corresponding samples from the rolling buffer into a synchronized clip. Each retained clip contains the selected camera payload, ego state, velocity, control commands, event labels, and window boundaries. For non-crash clips, INSIGHT applies persistent deduplication before committing the clip as canonical storage. If the new clip is sufficiently similar to a previously retained event, it is recorded as redundant metadata rather than stored again. Otherwise, it is written as a new event clip.

INSIGHT also maintains metadata as part of the runtime evidence. For each event, the metadata records trigger times, event classes, retained window bounds, clip size, deduplication outcome, and buffer status. This metadata supports later auditing and debugging: the stored trace shows what happened, while the metadata explains why the recorder retained or discarded the interval. The result is an online runtime that continuously observes vehicle behavior, preserves triggered evidence, and bounds storage growth without depending on any offline process.

B. Outcome-Driven Trigger Module

Current vehicles already provide mature collision sensing, and existing EDR mechanisms are designed around crash-event capture. We therefore treat crash as an established signal and focus the design discussion on the two less standardized triggers: near-crash risk and unstable-but-safe control.

1) *Near Crash Signal from Path Intersection TTC*: We define the near-crash signal S_{risk} from ego motion and tracked surrounding objects. The detector predicts short-horizon motion in the ego-centered ground plane and checks whether future occupied regions overlap under closing motion. Compared with distance-only screening, this suppresses nearby but non-conflicting objects. Figure 4 is presented to help with equation understanding.

a) *Inputs and prediction.*: At time t , the ego state is given by forward speed v_e and yaw rate ω_e . Each tracked object i provides planar position, velocity, and half extents:

$$\mathbf{p}_i(0) = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \quad \mathbf{v}_i = \begin{bmatrix} v_{x,i} \\ v_{y,i} \end{bmatrix}, \quad \mathbf{a}_i = \begin{bmatrix} a_{x,i} \\ a_{y,i} \end{bmatrix}. \quad (1)$$

Only objects inside a bounded monitoring region are evaluated:

$$-\rho_{rear} \leq x_i \leq \rho_{front}, \quad |y_i| \leq \rho_{lat}. \quad (2)$$

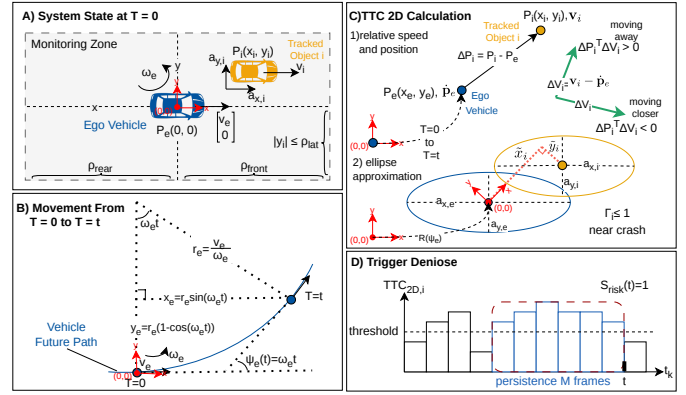


Fig. 4. Illustration of near-crash trigger. INSIGHT predicts ego and tracked object motion in the ego-centered frame and fires when sustained geometric overlap is predicted within the configured TTC threshold.

Under a constant speed and constant yaw rate model, the ego trajectory is

$$\mathbf{p}_e(t) = \begin{cases} \begin{bmatrix} v_e t \\ 0 \end{bmatrix}, & |\omega_e| \approx 0, \\ \begin{bmatrix} \frac{v_e}{\omega_e} \sin(\omega_e t) \\ \frac{v_e}{\omega_e} (1 - \cos(\omega_e t)) \end{bmatrix}, & \text{otherwise,} \end{cases} \quad (3)$$

with

$$\dot{\mathbf{p}}_e(t) = \begin{bmatrix} v_e \cos(\omega_e t) \\ v_e \sin(\omega_e t) \end{bmatrix}, \quad \psi_e(t) = \omega_e t. \quad (4)$$

Each object is propagated with a constant velocity model over horizon H :

$$\mathbf{p}_i(t) = \mathbf{p}_i(0) + t\mathbf{v}_i, \quad t_k = k\Delta t, \quad k = 1, \dots, K. \quad (5)$$

b) *Closing motion and overlap.*: For each future sample t_k , we compute

$$\Delta \mathbf{p}_i(t_k) = \mathbf{p}_i(t_k) - \mathbf{p}_e(t_k), \quad \Delta \mathbf{v}_i(t_k) = \mathbf{v}_i - \dot{\mathbf{p}}_e(t_k), \quad (6)$$

and the closing speed

$$c_i(t_k) = -\frac{\Delta \mathbf{p}_i(t_k)^T \Delta \mathbf{v}_i(t_k)}{\max(\|\Delta \mathbf{p}_i(t_k)\|_2, 1.0)}. \quad (7)$$

Object i is considered only if

$$c_i(t_k) \geq v_{close}^{min}. \quad (8)$$

Let $\mathbf{a}_e = [a_{x,e}, a_{y,e}]^T$ denote the ego half extents with safety padding. In the ego aligned frame,

$$\tilde{x}_i(t_k) = \cos \psi_e(t_k) \Delta x_i(t_k) + \sin \psi_e(t_k) \Delta y_i(t_k), \quad (9)$$

$$\tilde{y}_i(t_k) = -\sin \psi_e(t_k) \Delta x_i(t_k) + \cos \psi_e(t_k) \Delta y_i(t_k). \quad (10)$$

We define the normalized overlap score

$$\Gamma_i(t_k) = \left(\frac{\tilde{x}_i(t_k)}{a_{x,e} + a_{x,i}} \right)^2 + \left(\frac{\tilde{y}_i(t_k)}{a_{y,e} + a_{y,i}} \right)^2. \quad (11)$$

A geometric conflict occurs when

$$\Gamma_i(t_k) \leq 1. \quad (12)$$

c) *Final near crash trigger*.: The two dimensional time to collision for object i is

$$TTC_{2D,i} = \min \{t_k > 0 \mid c_i(t_k) \geq v_{close}^{min}, \Gamma_i(t_k) \leq 1\}, \quad (13)$$

with $TTC_{2D,i} = \infty$ if no such sample exists. The scene level TTC is

$$TTC_{2D} = \min_i TTC_{2D,i}. \quad (14)$$

The raw near-crash signal is

$$S_{risk}^{raw}(t) = \begin{cases} 1, & TTC_{2D}(t) < \tau_{risk}, \\ 0, & \text{otherwise,} \end{cases} \quad (15)$$

and the final trigger uses temporal persistence:

$$S_{risk}(t) = \mathbf{1} \left(\sum_{j=0}^{M-1} S_{risk}^{raw}(t-j) = M \right). \quad (16)$$

In the final system configuration, INSIGHT uses $\tau_{risk} = 1.5$ s and $M = 5$ frames.

2) *Unstable Control Signal*.: We define the unstable-control signal S_{ctrl} to capture difficult but non-crash episodes in which the final control stream exhibits repeated short-horizon corrections. The signal is computed directly from observable control commands and is therefore independent of model internals. Example signal is shown in Figure 5.

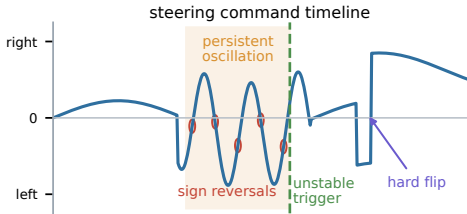


Fig. 5. Illustration of the unstable-control trigger.

At each control cycle, we form the normalized control vector

$$\mathbf{u}_t = \begin{bmatrix} \delta_t \\ b_t \\ a_t \end{bmatrix}, \quad (17)$$

where δ_t , b_t , and a_t denote steering, brake, and accelerator. Steering is mapped to $[-1, 1]$, and brake and accelerator are mapped to $[0, 1]$.

We compute the first and second order control differences:

$$\Delta \mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_{t-1}, \quad \mathbf{r}_t = \mathbf{u}_t - 2\mathbf{u}_{t-1} + \mathbf{u}_{t-2}. \quad (18)$$

Over a sliding window of length W_c , the Root Mean Square (RMS) control rate and RMS oscillation are

$$R_{rate}(t) = \sqrt{\frac{1}{W_c} \sum_{k=t-W_c+1}^t \|\Delta \mathbf{u}_k\|_2^2}, \quad (19)$$

$$R_{osc}(t) = \sqrt{\frac{1}{W_c} \sum_{k=t-W_c+1}^t \|\mathbf{r}_k\|_2^2}. \quad (20)$$

We normalize oscillation by recent command activity:

$$\rho_{ctrl}(t) = \frac{R_{osc}(t)}{R_{rate}(t) + \epsilon}, \quad (21)$$

and define the instability score

$$I_{ctrl}(t) = R_{rate}(t) \rho_{ctrl}(t) = \frac{R_{rate}(t) R_{osc}(t)}{R_{rate}(t) + \epsilon}. \quad (22)$$

This score is large only when the controller is both active and oscillatory, suppressing false triggers from nearly static commands and smooth aggressive maneuvers.

The final unstable control trigger requires persistence:

$$S_{ctrl}(t) = \mathbf{1} \left(\sum_{j=0}^{M_c-1} \mathbb{I}(I_{ctrl}(t-j) > \tau_{ctrl}) = M_c \right). \quad (23)$$

In the final system configuration, INSIGHT uses $\tau_{ctrl} = 0.8$ and $M_c = 5$ frames.

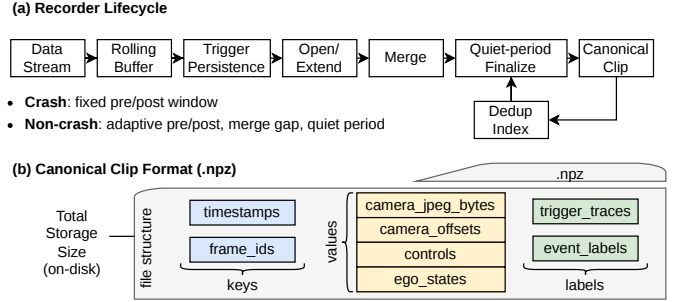


Fig. 6. Recorder lifecycle and canonical clip layout with raw byte streams.

C. Storage-Aware Recorder

The recorder converts trigger activations into evidence clips, following the lifecycle in Figure 6(a). During execution, INSIGHT maintains a bounded rolling buffer sampled at 10 Hz. Each trace sample stores timestamp, frame ID, encoded camera bytes, ego state, controls, detector outputs, and a serialized raw metadata payload containing IMU, surrounding objects, crash events, and non-image tick fields. The buffer length covers the largest required pre-event context.

a) *Window construction*.: When a crash event occurs, INSIGHT opens a crash window with fixed pre- and post-trigger bounds. When a near-crash or unstable-control trigger first becomes active, INSIGHT opens an adaptive non-crash window:

$$T_{pre} = \min(T_{pre}^{max}, T_{pre}^{min} + 0.5\Delta T), \quad (24)$$

$$T_{post} = \min(T_{post}^{max}, T_{post}^{min} + 0.5\Delta T), \quad (25)$$

where ΔT is the span between the first and latest trigger activation in the window. The current configuration uses 2–4 seconds of pre-trigger context and 1–2 seconds of post-trigger context. Additional activations extend the active window, and windows separated by less than the merge gap are merged so one driving episode is not fragmented.

b) *Quiet-period finalization.*: A non-crash window is not materialized immediately when the trigger drops. INSIGHT waits until the window end time has passed and the trigger has remained quiet for the configured quiet period. This prevents rapid near-crash or unstable bursts from producing many tiny clips. Crash windows finalize after their fixed post-trigger interval. At route end, any pending windows are force-finalized using the samples available in the buffer.

c) *Canonical clip format.*: When a window is ready, INSIGHT extracts all trace samples whose timestamps fall inside the window and writes the compressed .npz payload shown in Figure 6(b). Camera frames and raw per-sample metadata are stored as concatenated byte streams with offsets. The reported storage size is the actual on-disk size of this canonical clip file, including both compressed visual evidence and raw sample metadata.

D. Persistent Deduplication

Non-crash triggers can repeat across similar scenes or across nearby control oscillations. INSIGHT therefore performs persistent deduplication before accepting a non-crash clip as canonical. Deduplication is applied only to non-crash clips without crash labels; crash evidence is always retained.

Each candidate non-crash clip receives a signal fingerprint and visual hashes. The signal fingerprint resamples normalized local ego displacement, speed, yaw rate, steering, throttle, brake, trigger traces, duration, and event-class indicators into a fixed-length vector. Visual anchors are selected at the beginning, middle, and end of the clip; each anchor is decoded from JPEG and converted into a compact difference hash. A candidate is considered duplicate only if its normalized signal distance and visual Hamming distance are both below configured thresholds. Duplicate clips are recorded in metadata with their canonical ID, while canonical clips are written to disk and appended to a persistent deduplication index. This persistent index allows redundancy suppression across the whole run.

IV. EVALUATION AND EXPERIMENTS

A. Experimental Setup

a) *Platform, agents, and workload.*: We evaluate INSIGHT in closed-loop CARLA on Bench2Drive 220 [28], [29]. TF++ is the main driving agent because it provides a strong end-to-end driving baseline for this benchmark [30]. INSIGHT is attached to the scenario manager and receives route state online at each simulator tick. We also run TCP as a second driving agent to test whether the same outcome-driven recorder policy transfers to a different and weaker policy family [31].

The ego vehicle is instrumented with a front RGB camera, speedometer, GNSS, and IMU. In our recorder configuration, retained camera evidence is JPEG-compressed at 640×360 with quality 90 and sampled into the trace buffer at 10 Hz. Surrounding-object tracks are supplied by the simulator actor-state bridge used by the recorder runtime; in a vehicle deployment, these tracks correspond to fused objects from

an onboard perception or ADAS stack. For every route, we store the benchmark ground-truth annotations, INSIGHT route metadata, trigger events, recorded windows, canonical clips, and system metrics when running on the edge device.

b) *Storage accounting.*: Final INSIGHT storage is defined as the sum of actual on-disk sizes of canonical .npz clips. Each canonical clip stores compressed camera bytes, offsets for reconstructing frame payloads, ego pose and velocity, speed, GNSS, controls, detector outputs, event classes, window bounds, and raw per-sample metadata bytes. Crash-only EDR and continuous-recording baselines are estimated with the same measured bytes-per-second rate in CARLA simulation so that the comparison reflects a matched evidence format rather than different compression assumptions.

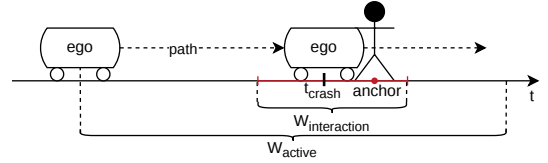


Fig. 7. Ground-truth window construction used in evaluation. Active windows cover the full scenario activation interval, while interaction windows narrow the label to ego engagement with a scenario-specific anchor.

c) *Ground truth and route scoring.*: Ground truth is collected online during each Bench2Drive route and saved in the per-route log, as illustrated in Figure 7. Crash labels are exact criterion-event timestamps reported by CARLA/ScenarioRunner. Non-crash scenarios have two complementary windows. Active windows mark the full period during which a Bench2Drive scenario is active. Interaction windows are narrower labels intended to capture ego engagement with the scenario's key anchor. Anchors are extracted from runtime state and scenario configuration, including trigger points, trigger locations, obstacle or collision locations, blocker waypoints, source/sink waypoints, and actor transforms. Interaction windows are opened when the ego reaches the anchor by distance or time-to-arrival.

We use the Bench2Drive composed score to summarize driving performance. Each route has route-completion score $S_{\text{route}} \in [0, 100]$, multiplicative infraction penalty $S_{\text{pen}} \in [0, 1]$, and final composed score

$$S_{\text{drive}} = S_{\text{route}} \times S_{\text{pen}}. \quad (26)$$

S_{route} is the percentage of the route completed before termination, while S_{pen} is reduced by benchmark infractions such as collisions, red-light violations, stop-sign violations, scenario timeout, and failure to yield.

d) *Baselines and comparison philosophy.*: We compare INSIGHT against crash-only EDR and continuous recording. Crash-only EDR captures the conventional evidence-retention endpoint. Continuous recording is the oracle endpoint for coverage, but pays full-route storage cost. INSIGHT is not expected to dominate continuous recording in coverage; its goal is to preserve more useful outcome evidence per byte while remaining deployable as an edge service.

To compare INSIGHT with a broader set of policies, we replay routes with complete streams under a matched retention budget. Baselines include periodic and random 4-s capture and a target-aware offline oracle; random results use 1,000 seeded trials. All policies use the same crash and scenario-window metrics.

B. Experiment 1: Threshold Sensitivity

To calibrate the triggers, we construct a deterministic 44-route development subset from a reference TF++ Bench2Drive run. The subset is selected to cover all scenario labels and rare safety events, with preference given to failed routes.

We then perform a grid search over TTC_{2D} thresholds $\{1.0, 1.5, 2.0\}$ and control instability thresholds $\{0.4, 0.6, 0.8\}$, while keeping all other detector and recorder parameters fixed. In particular, both temporal persistence settings are fixed at 5 frames. Each threshold pair is evaluated on the same 44-route subset using the TF++ agent. The goal is to find a threshold pair that gets the best capture-storage tradeoff.

C. Experiment 2: Event Capture Quality

This experiment asks how well the outcome-driven triggers capture more evidence beyond crash-only recording. We evaluate the TF++ Bench2Drive220 run by comparing INSIGHT outputs with Bench2Drive ground truth. INSIGHT outputs are evaluated at two stages: `recorded_windows`, which represent online selected windows after window construction, and `clips`, which represent final canonical stored clips after materialization and deduplication.

Crash recall is the fraction of ground-truth crash timestamps covered by at least one selected window or stored clip:

$$C_{\text{crash}}^{\text{det/clip}} = \frac{\#\{\text{GT crash ts hit by det/clip}\}}{\#\{\text{GT crash ts}\}}. \quad (27)$$

For non-crash scenarios, active-window coverage measures whether INSIGHT fires during meaningful scenario execution:

$$C_{\text{act}}^{\text{det/clip}} = \frac{\#\{\text{GT active win hit by det/clip}\}}{\#\{\text{GT active win}\}}, \quad (28)$$

and interaction-window coverage measures stricter anchor-centered timing:

$$C_{\text{inter}}^{\text{det/clip}} = \frac{\#\{\text{GT inter win hit by det/clip}\}}{\#\{\text{GT inter win}\}}. \quad (29)$$

We also report active-context precision, the fraction of INSIGHT outputs that overlap any active scenario window. Let $\mathcal{O}$ denote the selected windows or stored clips being evaluated, and let A denote the set of active scenario windows:

$$P_{\text{ctx}}^{\text{det/clip}} = \frac{\#\{o \in \mathcal{O} : o \cap A \neq \emptyset\}}{\#\mathcal{O}}. \quad (30)$$

Active-window coverage asks whether the recorder noticed the scenario context at all, or if it is a false positive that falls outside the baseline. Interaction-window coverage asks whether the trigger aligned with the benchmark’s narrow

anchor. Stored-clip metrics describe what remains after the storage deduplication policy.

To measure each trigger’s contribution, we retain crash and enable TTC, unstable control, or both. And then disable crash and measure whether the two non-crash triggers warn before collisions and by how long. Because INSIGHT receives object tracks from the perception stack, we also inject position/velocity noise, missed and spurious tracks, frame dropout, and latency to measure their effect on TTC triggering.

D. Experiment 3: Recording Efficiency

Recording efficiency evaluates the tradeoff between evidence coverage and retained bytes. The target set consists of crash evidence windows plus scenario interaction windows on routes with usable crash-plus-scenario accounting. We report both selected-window and stored-clip views for INSIGHT. Selected windows show the upper bound of what the online outcome policy identified. Stored clips show the stricter edge-realistic output after clip materialization and deduplication.

Storage is the total retained data of the final saved canonical clips. We further isolate deduplication by sweeping 24 signal/visual threshold pairs and inspecting the start, middle, and end of every removed pair. Measured byte rates are also projected to daily and annual writes and a 4 TB retention horizon under equal SSD endurance.

E. Experiment 4: Online Edge Deployability

We evaluate whether INSIGHT can execute continuously on a practical edge node under realistic online input streams to represent the real-world EDR device. The platform is shown in Figure 8. INSIGHT runs on a Raspberry Pi 5 with a Broadcom BCM2712 quad-core Arm Cortex-A76 CPU at 2.4 GHz, 8 GB memory, and a 256 GB SSD. CARLA, Bench2Drive220, and the driving agent run on the workstation, while the online tick stream is transmitted to the Pi over Ethernet. This setup represents INSIGHT as a separate edge recorder, decoupled from the main autonomy computer.

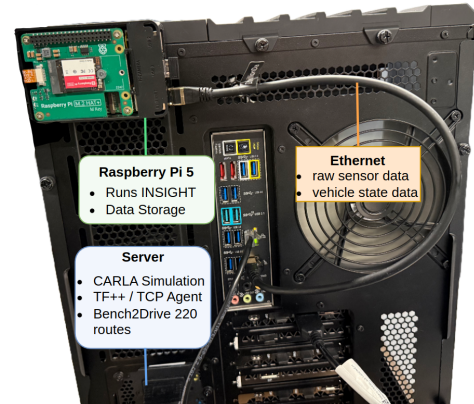


Fig. 8. The platform hardware connection of INSIGHT.

For each route, we measure detector latency, recorder latency, process CPU utilization, process RSS memory, system memory usage, and retained storage. Detector latency covers

ingestion and trigger update. Recorder latency covers buffer insertion, window update, merging, deduplication checks, and ready-window finalization. The TF++ and TCP deployments are executed sequentially so that resource measurements reflect one recorder workload at a time. The detector and recorder mean/P95/P99/maximum latency, including materialization and writes in the recorder path, are reported.

F. Experiment 5: Cross-Agent Transfer to TCP

We repeat the full pipeline with TCP using the same locked INSIGHT policy selected on the TF++ pilot subset. The goal is to test whether the recorder’s outcome policy remains meaningful for a different driving agent without retuning. We report two views. First, TCP driving performance is summarized over all 220 routes. Second, INSIGHT event capture, storage efficiency, and edge cost are evaluated on the subset of TCP routes that produce usable benchmark targets and recorder outputs. Because TCP is a weaker agent, the usable routes with ground truth are much smaller than the TF++ agent.

V. RESULTS AND ANALYSIS

A. Threshold Selection Results

Figure 9 shows a clear threshold tradeoff on the 44-route pilot subset. Lower TTC thresholds (1.0 s) keep storage small and can achieve high saved-clip precision, but they consistently underperform on interaction-window coverage, indicating that the detector becomes too conservative and misses relevant scenario evidence. Increasing the TTC threshold to 2.0 s improves coverage in some cases, but this gain comes with a larger storage footprint and reduced precision, suggesting that the more permissive trigger produces additional clips that are not consistently useful. Among all nine settings, TTC=1.5 s and unstable=0.8 provide the strongest overall operating point: it achieves the highest saved-clip interaction-window coverage (65.9%, tied for best), while also maintaining very high saved-clip precision (90.0%) and a relatively small final storage size (0.377 GB). In particular, it strictly outperforms the other tied high-coverage setting (TTC=2.0 s, unstable=0.4), which attains the same coverage but with substantially lower precision (79.3%) and higher storage (0.421 GB). We therefore select TTC=1.5 s and unstable=0.8 as the best pilot setting, since it offers the best balance between capturing interaction evidence, avoiding low-value saved clips, and keeping the final storage budget compact.

B. Event Capture Results

Table I summarizes event capture quality on the TF++ Bench2Drive 220-route run with threshold setting TTC = 1.5 s and unstable = 0.8. INSIGHT produces 419 online detection windows, which are further reduced to 156 final saved clips after clip generation and deduplication. Crash capture is reliable: all 77 ground-truth crash timestamps are covered by at least one detection window, and all 77 crash-labeled saved clips are correct, giving 100.0% crash detection recall and 100.0% crash clip precision. This suggests that once a crash-trigger

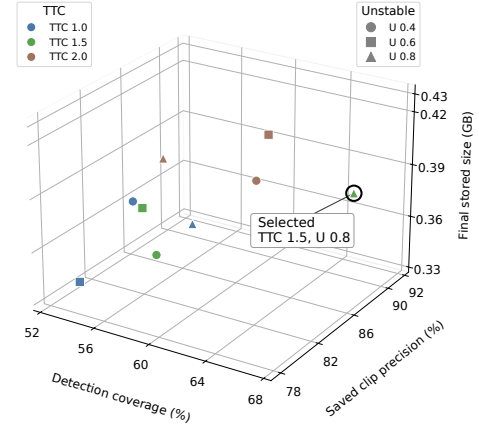


Fig. 9. Threshold sensitivity results of 9 threshold sets on a 44-route subset.

TABLE I
EVENT CAPTURE QUALITY ON THE TF++ BENCH2DRIVE 220-ROUTE RUN WITH THRESHOLD SETTING TTC = 1.5 s AND UNSTABLE = 0.8.

Metric	Value	Count
Detection windows generated	419	–
Final saved clips	156	–
Crash detection accuracy	100.0%	77/77
Crash saved clip precision	100.0%	77/77
Active-window detection coverage	80.5%	177/220
Active-window clip precision	98.1%	153/156
Interaction-window detection coverage	64.4%	145/225
Interaction-window clip precision	78.8%	123/156

condition occurs, the recorder reliably flushes and preserves the corresponding evidence.

For non-crash scenes, INSIGHT captures broad scenario context more consistently than the strict anchor-centered interaction interval. This is expected because INSIGHT is not a scenario-label detector. It fires when the agent outcome becomes abnormal, which can happen before the benchmark anchor opens, after the anchor closes, or not at all if the agent handles the scenario smoothly.

TABLE II
EVENT-CAPTURE BREAKDOWN FOR THE TF++ BENCH2DRIVE 220-ROUTE RUN. GT-SIDE ROWS REPORT COVERAGE OF BENCHMARK LABELS. INSIGHT-SIDE ROWS REPORT THE DISTRIBUTION OF INSIGHT OUTPUTS.

GT-side coverage				
Target	Total	Hit by det.	Hit by clips	Miss by det.
Crash events	77	77	77	0
Active windows	218	177	111	41
Interaction windows	225	145	81	80
INSIGHT-side distribution				
Output	Total	In inter.	Outside inter.	No active ctx.
Detection windows	419	227	189	3
Saved clips	156	123	30	3

a) *Interaction-window diagnostics.*: To understand the strict interaction-window misses, Table II separates a ground-truth coverage view from an INSIGHT-output overlap view.

On the GT side, all 41 missed active windows are complete no-trigger misses: each occurs on a route with zero INSIGHT detections. Those routes are usually easy for TF++, with a higher mean composed score than the active-hit subset (94.1 vs. 77.0), and 32 of the 41 missed-active routes even receive a perfect 100 score. This supports the core design claim: INSIGHT is linked to what the agent actually experiences, not just to the existence of a benchmark scenario label. If TF++ passes a scenario smoothly, staying silent is often the intended behavior.

The INSIGHT-side panel shows that most non-interaction outputs are not arbitrary false alarms. Only 3 of 419 detection windows and 3 of 156 saved clips fall outside GT active context, and all three are crash-related clips. One is an early crash on an Accident route whose collision happens before the GT scenario-active interval begins. The other two are crash routes that have usable crash GT but no usable scenario context.

TABLE III

DETECTION WINDOWS OUTSIDE THE STRICT INTERACTION WINDOW. EARLY AND LATE ARE DEFINED RELATIVE TO THE ANCHOR CENTERED INTERACTION INTERVAL. COUNTS AND PROPORTIONS ARE COMPUTED FROM THE 192 DETECTION WINDOWS THAT MISS THE STRICT INTERACTION WINDOW.

Metric	Early	Late
Count	70	122
Agent score profile	mean 88.8, median 100.0 78.6% success routes (55/70)	mean 43.6, median 13.5 68.9% failed routes (84/122)
Main trigger	near_crash 75.7% (53/70)	unstable 58.2% (71/122)

Table III breaks down the 192 detection windows that do not overlap the exact interaction window. These misses are not homogeneous. Early misses are dominated by good score routes and by *near_crash* triggers, which suggests that the agent often reacts slightly before the benchmark interaction window opens while the hazard is still developing and captured by INSIGHT. Late misses look different: they are concentrated on lower-score routes, which are dominated by *unstable* triggers, and they appear most often in roadside-hazard and parking-exit scenarios. This pattern reflects the agent’s performance and supports personalized data collection for future agents.

b) Scenario-family breakdown.: Figure 10 compares TF++ route performance with the average number of INSIGHT detection windows generated per route in each scenario family. The overall pattern is consistent with INSIGHT’s outcome-driven design: families with lower TF++ scores tend to produce more detection windows. Merge / actor flow has the highest mean score and the lowest trigger rate, with 1.00 window per route. In contrast, obstacle/blockage and other maneuver/control have the lowest mean scores and generate substantially more detections, with 2.40 and 3.20 windows per route, respectively. This indicates that INSIGHT remains relatively quiet when the agent handles the route smoothly, but becomes more active when the realized vehicle behavior includes crash, near-crash, or unstable-control outcomes.

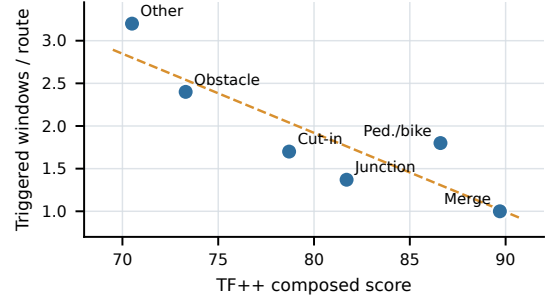


Fig. 10. Scenario-family outcome alignment. Each point is a scenario family, with TF++ composed score on the x-axis and mean INSIGHT detection windows per route on the y-axis.

Pedestrian and bicycle crossing is the main deviation from this monotonic trend. Although this family has a relatively high mean TF++ score, it still produces 1.80 triggered windows per route, exceeding both junction conflict and merge / actor flow. Route-level inspection shows that 19 of the 25 routes in this family receive good Bench2Drive scores, and these good-score routes account for 37 of the 45 triggered windows. The trajectory logs explain this behavior: many successful routes still contain aggressive close interactions with pedestrians or bicycles, so INSIGHT fires even when the benchmark score remains high. This is consistent with the recorder’s goal of retaining evidence of near-crash or difficult interaction behavior, not only failed routes.

This family also exposes a current limitation of the near-crash trigger. INSIGHT uses fixed safety bounds for tracked objects when estimating geometric conflict. This simple design is lightweight and conservative, but it can over-trigger for pedestrians and bicycles, whose effective safety regions should be smaller and more class-specific than vehicle-sized bounds. As a result, some high-score crossing routes generate additional near-crash windows. We keep this behavior in the evaluation because it reflects a practical tradeoff between conservative online retention and tighter object-specific calibration.

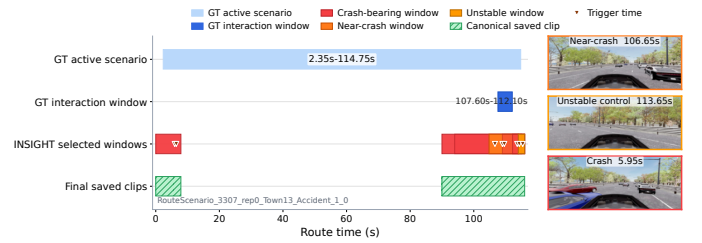


Fig. 11. Timeline case study for an Accident route (RouteScenario_3307_rep0_Town13). The right column shows the stored front-camera images at different trigger times.

c) Representative example.: Figure 11 illustrates the full INSIGHT workflow on an accident route with a composed score of 21.6. The route contains a long GT active scenario from 2.35s to 114.75s and a narrower GT interaction window from 107.60s to 112.10s. INSIGHT first records an early

crash-bearing window around the crash trigger at 5.95s, where the ego vehicle attempts to avoid a parked vehicle but collides with a vehicle approaching from the left rear. Later in the same route, the ego vehicle continues through the scenario and encounters additional risky interactions near the parked vehicles. INSIGHT then detects a near-crash at 106.65s, which overlaps the GT interaction window, and unstable-control behavior at 113.65s as the ego vehicle makes an aggressive avoidance maneuver after a sudden stop ahead.

The three front-camera snapshots in Figure 11 show the visual context at the corresponding trigger times: the early crash, the near-crash interaction near the parked vehicles, and the later unstable-control maneuver. The timeline also shows how these online trigger activations are converted into final retained evidence. Because the early crash window requires pre-crash context and the later near-crash and unstable-control windows occur close to the interaction interval, the recorder merges the relevant evidence into canonical saved clips rather than storing each trigger independently. The final retained clip covers the key interaction period around 89.65s–115.70s, preserving the 20s pre-crash context for the late crash-bearing evidence together with the subsequent near-crash and unstable-control behavior. This example shows how INSIGHT connects trigger-level observations to storage-level outputs: multiple outcome signals along one route are merged into compact retained evidence that captures the agent’s actual failure and recovery behavior.

Table IV compares matched-budget policies and summarizes trigger, tracking, storage, and latency sensitivity. At equal budget, INSIGHT improves strict coverage by 15.2 points over periodic capture and 23.3 points over random. TTC provides most non-crash coverage; unstable control adds three interactions. Tracking robustness decreases with input quality.

C. Storage Efficiency Results

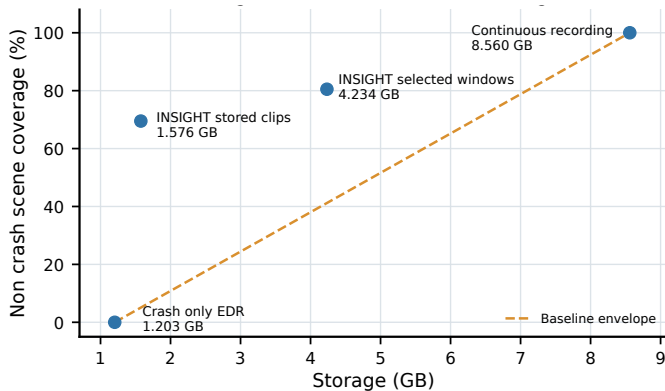


Fig. 12. Storage-coverage comparison for selective recording. INSIGHT is evaluated both as online selected windows and as final stored clips after the storage policy.

Figure 12 compares the storage cost required to retain non-crash scene evidence. The stored-clip result shows the effect of the complete recorder. INSIGHT preserves 69.5% of non-crash scene coverage using 1.576 GB, while continuous

recording requires 8.560 GB to reach full coverage of 220 routes. Compared with crash-only EDR, INSIGHT adds only 0.373 GB but recovers a large portion of non-crash evidence. This result indicates that useful non-crash data is concentrated in a small set of triggered intervals, rather than spread evenly across the full driving stream.

Before final filtering, selected windows cover 80.5% using 4.234 GB; merging, materialization, and deduplication reduce storage by 62.8%, while stored-clip coverage becomes 69.5%. This change reflects the complete pipeline, not deduplication alone: at 0.75/18, deduplication removes seven event- and scenario-matched clips without coverage loss, whereas aggressive settings lose evidence (Table IV). By concentrating storage on triggered intervals, INSIGHT remains above the uniform baseline envelope.

These results show that INSIGHT improves the storage-to-coverage tradeoff for more event evidence. It provides a practical edge-recording policy that captures high-value non-crash events while keeping onboard storage cost low.

D. Edge Level System Performance

The detector and recorder average 0.89 and 1.49 ms, respectively, so their 2.38 ms combined path uses 2.38% of each 100 ms cycle. Figure 13 reports mean/P95 latency, while Table IV adds P99/maximum and shows that materialization and SSD writes create the recorder tail.

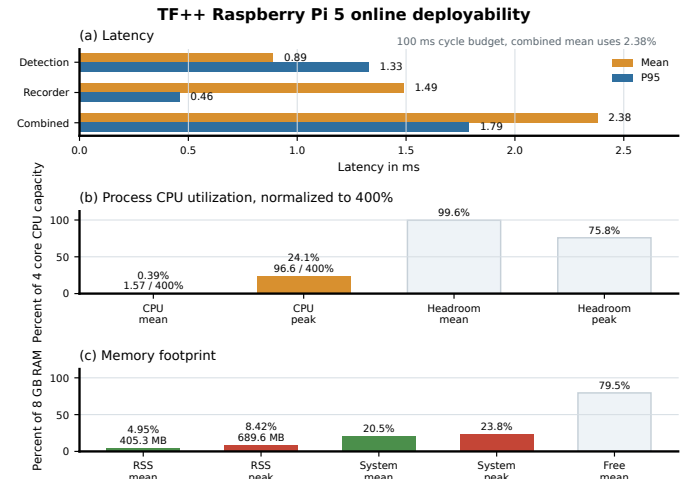


Fig. 13. Raspberry Pi 5 deployment dashboard with TF++ agent. INSIGHT uses a small fraction of the 100 ms cycle budget, low average CPU, and bounded memory footprint in the TF++ Bench2Drive220 run. CPU utilization is normalized against the 4 core Raspberry Pi 5 process capacity.

The CPU profile is bursty rather than sustained. Mean process CPU utilization is 1.57%, while the peak reaches 96.6%. On Raspberry Pi 5, which exposes four CPU cores, a process can reach up to 400% CPU utilization under the common Linux process accounting convention. The observed peak therefore corresponds to roughly one core during short bursts, not saturation of the platform. These bursts are expected when event windows are opened, clip data is materialized, compressed, checked for redundancy, or flushed to storage.

TABLE IV

COVERAGE, STORAGE, AND LATENCY SENSITIVITY. MATCHED-POLICY AND TRACKING ROWS USE 220 ROUTES (62 CRASHES, 223 INTERACTIONS, 444 CANDIDATES); TRACK SETTINGS LIST POSITION/VELOCITY NOISE, MISS RATE, AND LATENCY. BRACKETS SHOW 95% INTERVALS.

Evaluation	Policy or setting	Main result	Complementary result
Matched budget	INSIGHT	176/223 strict (78.9%)	62/62 crashes; 2,641 s retained
	Periodic 4 s	142/223 strict (63.7%)	27/62 crashes; 2,640 s retained
	Random 4 s (1,000 trials)	55.6% [50.7–61.0%] strict	42.4% mean crash coverage
	Offline oracle	223/223 strict (100%)	62/62 crashes; 5,683 s acquired
Trigger ablation	Crash+(TTC / unstable / both)	173 / 40 / 176 of 223 strict	–
	Pre-impact, no crash trigger	36/62, 58/62, 60/62 within 1/3/10 s	1.58 s median lead
Track perturbation	Mild: 0.25 m/0.25 m/s, 5%, 100 ms	85.6% [83.8–87.2%] retained	0.37 s delay
	Moderate: 0.5 m/0.5 m/s, 10%, 200 ms	66.2% [62.1–67.9%] retained	0.76 s delay
	Severe: 1 m/1 m/s, 20%, 500 ms	36.6% [34.5–38.6%] retained	1.69 s delay
Deduplication	None	0 removed; 0% bytes	176/223 strict; 202/217 active
	Signal/visual 0.75/18	7 removed; 1.2% bytes	62/62 crash; 176/223 strict; 202/217 active
	Signal/visual 0.75/48	109 removed; 19.0% bytes	155/223 strict; 188/217 active; 63/109 same type
Latency tail	Detector	Mean/P95/P99/max: 0.89/1.33/1.33/2.62 ms	<100 ms
	Recorder	Mean/P95/P99/max: 1.49/0.46/8.86/60.56 ms	Materialization/write tail
4-TB projection	Continuous / INSIGHT	127.1 / 23.4 GB/day; 31.5 / 171 days	46.4 / 8.54 TB/year; 5.43 × life

The low mean CPU shows that these expensive operations are infrequent relative to the normal per tick trigger and buffer update path.

Memory usage remains modest for an 8 GB edge device. The process RSS is 405.3 MB on average and peaks at 689.6 MB, corresponding to about 4.95% and 8.42% of device memory, respectively. System memory usage remains between 20.5% and 23.8%. This leaves substantial memory headroom for the operating system, networking stack, and other vehicle services. The larger RSS compared with the earlier run reflects the updated recorder configuration and larger retained clip accounting, but the footprint is still well within the practical range of a Raspberry Pi 5 class platform. The smaller difference between the mean and peak also shows the stable and bounded memory usage.

Together, the timing slack, low mean CPU, brief one-core bursts, and bounded memory support continuous 10 Hz execution of the external recorder, demonstrating its edge-deployment capabilities in the real world.

E. Transfer Study with TCP Agent

TCP completed only 22 of 220 routes, with a mean Bench2Drive composed score of 9.1. This weak performance sharply reduces the amount of usable TCP ground truth: the TCP run contains only 42 active windows, 38 interaction windows, and 86 storage-analysis targets.

TABLE V
TCP EVENT-CAPTURE QUALITY ON THE USABLE TCP SUBSET.

Metric	Value	Count
Detection windows generated	86	–
Final saved clips	86	–
Crash detection and save accuracy	100.0%	48/48
Active window coverage	90.5%	38/42
Interaction window coverage	76.3%	29/38

Within that reduced subset, the locked INSIGHT policy still transfers reasonably well. Table V shows that crash

capture stays perfect at 48/48, active-scenario capture reaches 38/42 (90.5%), and interaction-window recall reaches 29/38 (76.3%). The key finding is that TCP often fails before a benchmark interaction window fully develops, but when it remains engaged long enough to create a usable target, the TF++-selected thresholds still fire at the right time. In this sense, transferability here means that the same detector policy continues to identify relevant behavior under a different and much weaker driving stack.

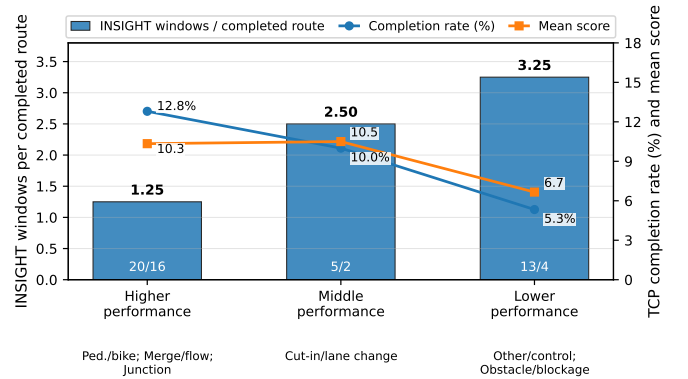


Fig. 14. TCP cross-agent transfer by performance group. Families are grouped by TCP completion and score to reduce small-denominator noise. Bars show INSIGHT windows per completed route, and lines show TCP completion rate and mean score. Lower TCP performance leads to denser INSIGHT detections.

Figure 14 summarizes the TCP cross-agent transfer result by grouping scenario families according to route-level performance. Because TCP is a much weaker agent than TF++, many routes terminate early and only a small number of completed routes remain in some families. Directly comparing each family can therefore introduce noise from very small denominators. To reduce this effect, we group families by TCP completion and score, and report INSIGHT detected windows normalized by the number of completed routes.

The grouped result shows that INSIGHT detection den-

sity is still linked to agent performance under TCP. The higher-performance group has the highest completion rate and relatively higher score, and produces only 1.25 INSIGHT windows per completed route. The middle-performance group produces 2.50 windows per completed route, while the lower-performance group produces 3.25 windows per completed route. This increasing trend indicates that when TCP performs worse, the completed executions contain denser near-crash or unstable-control behavior, and INSIGHT correspondingly generates more event windows.

This result supports the transferability of INSIGHT to a different driving agent. The same INSIGHT policy selected on TF++ remains meaningful for TCP without agent-specific re-tuning: even though TCP completes fewer routes and has lower overall driving performance, INSIGHT still detects outcome-driven windows that reflect the agent’s realized behavior.

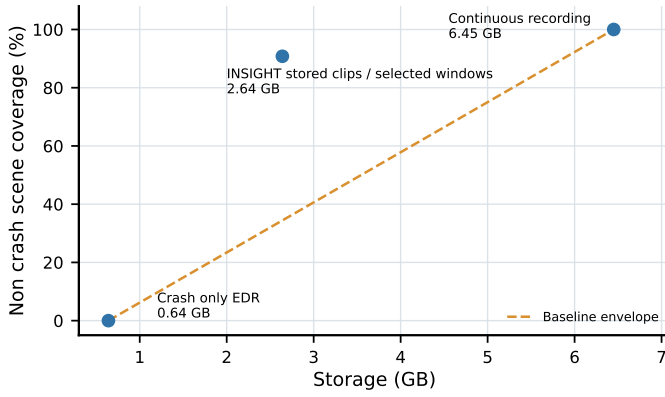


Fig. 15. TCP storage-coverage comparison for selective recording. There is no window deduplication.

The storage story also changes in an informative way. Figure 15 shows that INSIGHT stored clips cover 90.8% of active-window scenarios, indicating that the weaker TCP agent triggers INSIGHT frequently inside safety active windows. The final stored clips require 2.64 GB, which is above the crash-only baseline but still far below continuous recording, showing a favorable storage and event-capture tradeoff. An interesting observation is that no selected window is removed by deduplication. This is mainly because TCP completes only 22 routes, while most failed routes terminate early. Therefore, there are fewer repeated events and repeated vehicle behaviors across the run. As a result, every detected crash or non-crash window is saved as a final stored clip.

Figure 16 shows that INSIGHT maintains the same lightweight edge behavior under the TCP agent. Despite TCP’s weaker driving performance, the online detector and recorder remain comfortably within the real-time execution budget, while CPU demand stays low on average and memory usage remains bounded. This suggests that the system overhead is mainly determined by INSIGHT’s trigger and recording logic rather than by a specific driving agent. Overall, the TCP deployment confirms that INSIGHT can generalize across agents while remaining practical for low-cost vehicle-edge

platforms such as the Raspberry Pi 5.

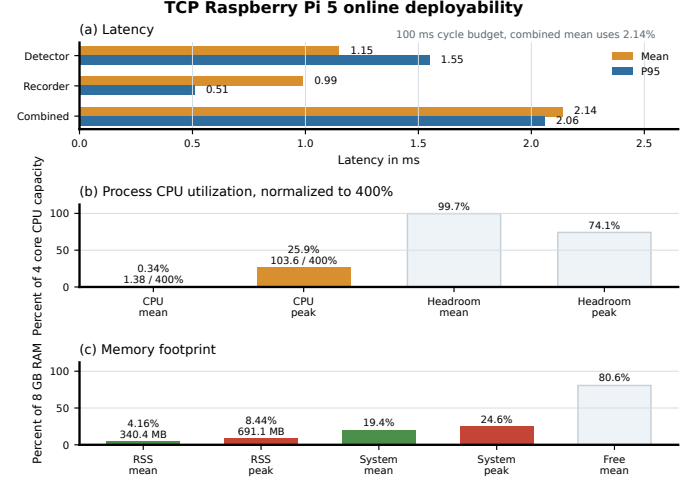


Fig. 16. Raspberry Pi 5 deployment dashboard with TCP agent. INSIGHT uses a small fraction of the 100 ms cycle budget, low average CPU, and bounded memory footprint in the TCP Bench2Drive220 run. CPU utilization is normalized against the 4 core Raspberry Pi 5 process capacity.

VI. CONCLUSION AND FUTURE WORK

We presented **INSIGHT**, an outcome-driven event recorder that retains evidence when the deployed agent crashes, approaches a collision, or exhibits unstable control. Its online triggers and storage-aware recorder capture valuable non-crash events missed by crash-only EDR while avoiding the cost of continuous recording.

On 220 TF++ routes, INSIGHT preserves all 77 crashes and 69.5% of non-crash evidence using 1.576 GB, 81.6% less storage than continuous recording, with 0.39% mean CPU, 24.1% peak CPU, and 405.3/689.6 MB mean/peak RSS. This demonstrates that INSIGHT retains more high-value evidence within a limited storage budget. On Raspberry Pi 5, its 2.38 ms mean latency fits within the 100 ms cycle, supporting real-time edge execution. Without retuning, INSIGHT also transfers to TCP, preserving 48/48 crashes, 38/42 active windows, and 29/38 interaction windows, and reducing storage by 59.1% compared with continuous recording. The TCP edge deployment remains lightweight, with 2.14 ms combined mean latency.

INSIGHT also exposes open research questions. Class-independent risk bounds can over-trigger for vulnerable road users, simulation-derived tracks underrepresent real perception uncertainty, and low TCP completion limits the transfer evidence. Future work can develop uncertainty- and class-aware triggers, validate outcome-driven retention across diverse agents and real-vehicle deployments, and study long-term storage behavior, extending INSIGHT toward a general platform for agent-aware data collection.

ACKNOWLEDGMENT

This work is funded by the U.S. National Science Foundation (NSF) under Grant CNS 2231523. Special thanks are extended to Western Digital for their support.

REFERENCES

- [1] Uber Technologies, Inc., “Uber AV: Autonomous Mobility and Delivery,” <https://www.uber.com/us/en/autonomous/>, 2026, accessed: 2026-04-30.
- [2] Nuro, Inc., “Safety,” <https://www.nuro.ai/safety>, 2026, accessed: 2026-04-30.
- [3] U.S. Department of Transportation, “Event data recorders,” Federal Register final rule summary, 2024, final rule published December 18, 2024. Accessed 2026 04 13. [Online]. Available: <https://www.transportation.gov/regulations/federal-register-documents/2024-29862>
- [4] Garmin, “Dash Cam Recording,” <https://www8.garmin.com/manuals/webhelp/GUID-05C32B20-1E3D-4193-88A0-507B33C3434E/EN-US/GUID-39BB7445-FF9C-4CC2-89AF-07311CAF07C9.html>, 2026, accessed: 2026-04-30.
- [5] D. Bogdoll, M. Nitsche, and J. M. Zöllner, “Anomaly detection in autonomous driving: A survey,” *arXiv preprint arXiv:2204.07974*, 2022.
- [6] S. Rahmani *et al.*, “A systematic review of edge case detection in automated driving: Methods, challenges and future directions,” *arXiv preprint arXiv:2410.08491*, 2024.
- [7] J. Yang *et al.*, “OpenOOD: Benchmarking generalized out-of-distribution detection,” in *Advances in Neural Information Processing Systems*, 2022.
- [8] National Highway Traffic Safety Administration, “49 CFR Part 563 event data recorders,” Electronic Code of Federal Regulations, 2025, accessed 2026 04 13. [Online]. Available: <https://www.ecfr.gov/current/title-49/subtitle-B/chapter-V/part-563>
- [9] J. C. Hayward, “Near-miss determination through use of a scale of danger,” *Highway Research Record*, vol. 384, pp. 24–34, 1972.
- [10] B. L. Allen, B. T. Shin, and P. J. Cooper, “Analysis of traffic conflicts and collisions,” *Transportation Research Record*, vol. 667, pp. 67–74, 1978.
- [11] F. Guo, S. G. Klauer, M. T. McGill, and T. A. Dingus, “Evaluating the relationship between near-crashes and crashes: Can near-crashes serve as a surrogate safety metric for crashes?” National Highway Traffic Safety Administration, Tech. Rep. DOT HS 811 382, 2010. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/dot_hs_811_382.pdf
- [12] Y. Wang, Y. He, R. Wang, and W. Shi, “Quantitative analysis of storage requirement for autonomous vehicles,” in *Proceedings of the 16th ACM Workshop on Hot Topics in Storage and File Systems*, 2024, pp. 71–78.
- [13] Y. Yao and E. M. Atkins, “The smart black box: A value driven high bandwidth automotive event data recorder,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1484–1496, 2021.
- [14] R. Feng, Y. Yao, and E. Atkins, “Smart black box 2.0: Efficient high bandwidth driving data collection based on video anomalies,” *Algorithms*, vol. 14, no. 2, p. 57, 2021.
- [15] K. Li *et al.*, “CODA: A real-world road corner case dataset for object detection in autonomous driving,” in *Computer Vision – ECCV 2022*, 2022.
- [16] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [17] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, “Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 948–10 957.
- [18] Y. Sun and Y. Li, “DICE: Leveraging sparsification for out-of-distribution detection,” in *Computer Vision – ECCV 2022*, ser. Lecture Notes in Computer Science, vol. 13684. Springer, 2022, pp. 691–708.
- [19] T. Vojř, T. Šipka, R. Aljundi, N. Chumerin, D. O. Reino, and J. Matas, “Road anomaly detection by partial image reconstruction with segmentation coupling,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 651–15 660.
- [20] K. Lis, S. Honari, P. Fua, and M. Salzmann, “Detecting road obstacles by erasing them,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 4, pp. 2518–2534, 2024.
- [21] Z. Hu, J. Shen, S. Guo, X. Zhang, Z. Zhong, Q. A. Chen, and K. Li, “Pass: A system-driven evaluation platform for autonomous driving safety and security,” in *NDSS Workshop on Automotive and Autonomous Vehicle Security*, 2022. [Online]. Available: <https://par.nsf.gov/biblio/10359464>
- [22] S. P. Venthuruthiyil and M. C. Chunchu, “Anticipated collision time (ACT): A two-dimensional surrogate safety indicator for trajectory-based proactive safety assessment,” *Transportation Research Part C: Emerging Technologies*, vol. 139, p. 103655, 2022.
- [23] O. A. Osman, M. Hajji, P. R. Bakhit, and S. M. Ishak, “Prediction of near-crashes from observed vehicle kinematics using machine learning,” *Transportation Research Record*, vol. 2673, no. 9, pp. 384–393, 2019.
- [24] H. Guo, K. Xie, and M. Keyvan-Ekbatani, “Modeling driver’s evasive behavior during safety-critical lane changes: Two-dimensional time-to-collision and deep reinforcement learning,” *Accident Analysis & Prevention*, vol. 186, p. 107063, 2023.
- [25] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, and E. Böde, “Criticality metrics for automated driving: A review and suitability analysis of the state of the art,” *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 1–35, 2023.
- [26] L. Westhofen, C. Neurohr, M. Butz, M. Scholtes, and M. Schuldes, “Using ontologies for the formalization and recognition of criticality for automated driving,” *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 214–230, 2022.
- [27] K. Shimomura *et al.*, “OD-RASE: Ontology-driven risk assessment and safety enhancement for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [28] X. Jia, Z. Yang, Q. Li, Z. Zhang, and J. Yan, “Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving,” in *NeurIPS*, 2024.
- [29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in *CoRL*, 2017.
- [30] B. Jaeger *et al.*, “Hidden biases of end-to-end driving models,” in *ICCV*, 2023.
- [31] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, “Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline,” in *Advances in Neural Information Processing Systems*, 2022.